



F r o m T e c h n o l o g i e s t o S o l u t i o n s

SQL Server Integration Services Using Visual Studio 2005

A Beginners Guide

Jayaram Krishnaswamy

[PACKT]
PUBLISHING

SQL Server Integration Services Using Visual Studio 2005

A Beginners Guide

Jayaram Krishnaswamy



BIRMINGHAM - MUMBAI

SQL Server Integration Services Using Visual Studio 2005

A Beginners Guide

Copyright © 2007 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, Packt Publishing, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: December 2007

Production Reference: 1171207

Published by Packt Publishing Ltd.
32 Lincoln Road
Olton
Birmingham, B27 6PA, UK.

ISBN 978-1-847193-31-5

www.packtpub.com

Cover Image by Nikhil Bangera (nikh@minldless.com)

Credits

Author	Project Manager
Jayaram Krishnaswamy	Patricia Weir
Reviewer	Project Coordinator
Anand Narayanaswamy	Abhijeet Deobhakta
Senior Acquisition Editor	Indexer
Douglas Paterson	Monica Ajmera
Development Editor	Proofreader
Rashmi Phadnis	Cathy Cumberlidge
Technical Editor(s)	Production Coordinator
Ajay.S	Aparna Bhagat
Sarvesh Shanbag	Shantanu Zagade
Editorial Team Leader	Cover Designer
Mithil Kulkarni	Aparna Bhagat

About the Author

Jayaram Krishnaswamy has been working in IT related fields since 1997. He was once a Microsoft Certified Trainer in Networking and a Siebel developer. He has worked with several IT related companies, such as Butler International in their Siebel practice; several other IBM sub-contractors, and smaller companies.

At present, he is active in writing technical articles in the IT field to many online sites such as Code Project.com, ASPFree.com, DevShed.com, DevArticles.com, OfficeUsers.com, AspAlliance.com, and many others. During 2006-2007 he wrote more than 200 articles, mostly related to database and web-related technologies covering Microsoft, Oracle, Sybase, ColdFusion, Sun, and other vendor products.

Acknowledgements

First of all, I would like to thank Dr. Douglas Paterson, the Senior Acquisition Editor, for encouraging me to write this book and giving his time generously. When I was toying with the idea of combining DTS and SSIS in one book, he persuaded me to stick with SSIS. I am most grateful for this suggestion as I can see now that the book is better focused. He was also instrumental in guiding me when I was writing the initial chapters that form the foundation for the rest of the book.

I would like to thank Abhijeet Deobakta, the Project Manager, for the way he distributed the authoring tasks, followed by the smooth and steady flow of work during the revisions. I thank Zenab Kapasi who was involved in an earlier phase of coordinating the authoring process. I thank Rashmi Phadnis, the Development Editor, for her suggestions as well as the contribution she made during the revisions. I would also like to thank Patricia Weir, the Project Manager, who helped me both during and after the contract phase.

I acknowledge and thank the different folks who reviewed the book. I sincerely thank the reviewers Douglas Paterson, Anand Narayanaswamy (Microsoft Most Valuable Professional (MVP) in Visual C#), and Rashmi Phadnis. The value added to the book by the reviewers is really enormous. As most of the chapters of the book depend on the reader taking the steps detailed by the author, it becomes a difficult task for the reviewer as he/she has to make sure the author has not erred in laying the steps. I sincerely thank the reviewers for the meticulous review.

I would like to thank my parents who would have very much shared my joy. I thank my brothers and sisters, and the whole hearted support of our Subbagiri family. Last but not least, I cannot sufficiently thank my wife, Michiko Fukumoto, without whose support and encouragement this work would have been impossible.

About the Reviewers

Anand Narayanaswamy works as an independent consultant and runs NetAns Technologies (www.netans.com), which provides web hosting services and is based in Trivandrum, India. Anand is a Microsoft Most Valuable Professional (MVP) in Visual C# (<https://mvp.support.microsoft.com/profile/Anand>) and is the author of *Community Server Quickly* (<http://www.packtpub.com/community-server/book>) published by Packt Publishing.

He works as the chief technical editor for ASPAlliance.com (<http://aspalliance.com/author.aspx?uId=384030>) and is also a member of ASPAlliance.com Advisory Board. He regularly contributes articles, and book and product reviews to ASPAlliance.com, C-Sharpcorner.com, Developer.com, Codeguru.com, Microsoft Academic Alliance, and asp.netPRO magazine.

Anand has worked as a technical editor for several popular publishers such as Sams, Addison-Wesley Professional, Wrox, Deitel, and Manning. His technical editing skills helped the authors of Sams Teach Yourself the C# Language in 21 Days, Core C# and .NET, Professional ADO.NET 2, ASP.NET 2.0 Web Parts in Action and Internet and World Wide Web (4th Edition) to fine tune the content. He has also contributed articles for Microsoft Knowledge Base and delivered podcast shows for Aspnetpodcast.com. He is a moderator for Windows MarketPlace Newsgroups.

Anand also runs LearnXpress.com (www.learnxpress.com), Dotnetalbum.com (www.dotnetalbum.com), CsharpFAQ.com (www.csharpfaq.com) and Devreviews.com (www.devreviews.com). LearnXpress.com is a featured site at MSDN's Visual C# .NET communities section. Anand has won several prizes at Community-Credit.com and has been featured as "All Time" contributor at the site. He is one of the founders of Trivandrum Microsoft Usergroup. He regularly blogs under the banner "I type what I feel" at <http://msmvps.com/blogs/anandn> and maintains a personal website at www.visualanand.net.

Table of Contents

Preface	1
Chapter 1: SSIS Basics	7
SQL Server Integration Services	7
Objects Used in SSIS	9
The SSIS Package	10
The Control Flow Elements	10
Data Flow Components	12
Data Source Components	12
Data Transformation	14
Data Flow Destinations	21
Connection Managers	22
Variables	24
Event Handlers	24
Log Providers	25
Debugging and Diagnostic Features	26
Summary	26
Chapter 2: Creating a BI Project for SSIS in Visual Studio 2005	27
Business Intelligence using Microsoft Products	27
Resources Used for Creating Projects	28
Creating Your First BI Project for SSIS	28
Launching VS 2005 and Creating a BI Project for Integration Services	28
Business Intelligence Project Properties	32
Overview of the Project Window	33
Canvas for Package Design	37
Control Flow	38
Data Flow	39
Event Handlers	41
Package Explorer	43
The Toolbox	44

Table of Contents

The Solution Explorer	48
Getting Various Windows	53
Server Explorer Window	54
Bookmark Window	55
Class View and Code Definition Windows	56
Object Browser	56
Error List Window	58
Output Window	58
Properties Window	58
Tasks List Window	60
Toolbox Window	60
Find Results Window	60
Other Windows	63
Debug Windows	65
BI Related Items in Tools/Options	66
Property Pages and Folders of the Project	67
Executing the Package and Saving the Project	68
Hands-On Exercises	69
Hands-On Exercise 1	69
Hands-On Exercise 2	69
Hands-On Exercise 3	72
Summary	73
Chapter 3: Sending Email with a SSIS Package	75
Hands-On Exercise One: Sending an Email Using the SMTP Server	75
Hands-On Exercise Two: How to Find Your ISP's SMTP Server?	84
Summary	85
Chapter 4: Transferring Data to a Text File	87
Hands-On Exercise: Transferring Data to a Text File	87
Step 1: Creating a BI Project and Adding a Data Flow Task	88
Step 2: Adding Connection Manager for the DataReader	88
Step 3: Configuring the Source	90
Step 4: Adding a Flat File Destination and Establishing a Path from DataReader Source	93
Step 5: Configuring the Flat File Destination Component	94
Step 6: Build and Execute the Package	97
Summary	98
Chapter 5: Transferring Data to a Microsoft Excel File	99
Hands-On Exercise: Transferring Data to an Excel File	99
Step 1: Creating a BI Project and Adding a Data Flow Task	100
Step 2: Configuring the DataReader's Connection Manager	100
Step 3: Configuring the DataReader Source	101
Step 4: Adding a Character Map Transformation	102

Step 5: Adding an Excel Destination and Establishing a Path to It from the Character Map Data Flow Component	106
Step 6: Configuring the Excel Destination Component	107
Step 7: Testing the Package	110
Summary	110
Chapter 6: Data Transfer to an MS Access Database	111
Hands-On Exercise: Transferring Data to an Access Database	111
Step 1: Creating a BI Project and Adding a Data Flow Task	112
Step 2: Configuring the DataReader's Connection Manager	112
Step 3: Configuring the DataReader Source	112
Step 4: Adding an OLE DB Destination and Establishing a Path from the DataReader Component	113
Step 5: Configuring the OLE DB Destination Component	114
Step 6: Incorporating a Data Viewer to Monitor Data Flow	119
Summary	122
Chapter 7: Data Transfer from a Text File Using the Bulk Insert Task	123
Hands-On Exercise: Transferring Data from a Flat File to a SQL Server Database Table	123
Step 1: Use / create a Flat Text File whose Contents Need to be Transferred	124
Step 2: Create a Table with Columns that Can Accept the Contents of the File Created	124
Step 3: Create a BI Project and Add a Bulk Insert Task	126
Step 4: Configure the Bulk Insert Task	126
Step 5: Build and Execute the Package	131
What Happens if there Is an Error?	132
Summary	133
Chapter 8: Using a Conditional Split Data Transformation	135
Hands-On Exercise: Splitting Data Retrieved from a SQL Server	135
Step 1: Create a BI Project and Add a Data Flow Task. Add and Configure the DataReader Source to Pull Data from the Local SQL Server	136
Step 2: Add a Conditional Split Transformation	137
Step 3: Establish a Path to Connect DataReader Source with the Conditional Split Data Transformation	137
Step 4: Configure the Conditional Split Data Transformation	137
Step 5: Add Recordset Destination(s)	140
Step 6: Configure the Recordset Destination(s)	141
Step 7: Build, Execute the Package and Review	145
Summary	145

Table of Contents

Chapter 9: Using an Aggregate Data Transformation	147
Hands-On Exercise: Using Aggregate Data Flow Transformation	148
Step 1: Create a BI Project and Add a Data Flow Task. Add and Configure the DataReader Source to Pull Data from the Local SQL Server	149
Step 2: Add an Aggregate Data Transformation	149
Step 3: Establish a Path to Connect DataReader Source with the Aggregate Data Transformation	150
Step 4: Configure the Aggregate Data Flow Transformation	150
Step 5: Add a Percentage Sampling Data Transformation	153
Step 6: Establish a Path from Aggregate Data Transformation to the Percentage Sampling Data Transformation	153
Step 7: Configure the Percentage Sampling Data Flow Item	154
Step 8: Add a Recordset Destination Data Flow Component	155
Step 9: Configure the Recordset Destination Data Flow Component	155
Step 10: Build and Execute the Package, and Review Results	156
Summary	158
Chapter 10: Using a Data Conversion Data Flow Transformation	159
Source and Destination for the Exercise	159
Hands-On Exercise: Transferring Data to an Excel File	162
Step 1: Creating a BI Project; Adding a Data Flow Task and Configuring an Excel Source	162
Step 2: Adding a Data Conversion Data Flow Transformation	165
Step 3: Establishing a Path from Excel Source to Data Conversion Data Flow Transformation	166
Step 4: Configuring the Data Conversion Data Transformation	167
Step 5: Adding and Configuring an OLE DB Data Destination	168
Adding an OLE DB Destination to Canvas	168
Establish a Path from Data Conversion Data Flow Transformation to OLE DB Data Destination	168
Setting Up a Connection Manager	169
Displaying the Table to which the Data will be Inserted	170
Column Mappings	171
Handling Data Loading Errors	171
Step 6: Adding and Configuring a Recordset Destination for Displaying Errors	172
Step 7: Building the Project and Testing the Package	173
Summary	174
Chapter 11: Creating a SSIS Package with an XML Task	175
Diff	175
Merge	176

Validate	176
XPATH	176
XSLT	176
XML Documents Used in This Chapter	176
Documents Used for Diff XMLTask Type	177
Documents Used for XSLT XMLTask Type	178
Hands-On Exercise 1: XMLTask type Diff	179
Step 1: Create a BI Project and Add a Control Flow task, XMLTask	179
Step 2: Configuring XMLTask	179
Hands-On Exercise 2: XMLTask Type XSLT	184
Summary	186
Chapter 12: Creating a SSIS Package to Access Folders and Files	187
Hands-On Exercise: Copying a File from One Folder to Another and Sending it using the Send Mail Task	188
Step 1: Creating a BI Project and Adding a Control Flow Task — the File System Task	188
Step 2: Configuring The File System Task	189
Step 3: Adding a Send Mail Task to the Control Flow Page on the Canvas	192
Using Precedence Constraint to Send Mail	195
Step 4: Adding a Precedence Constraint	196
Step 5: Building and Executing the Package	197
Summary	197
Chapter 13: Package to Copy a Table from Oracle XE	199
Hands-On Exercise: Transferring a View from Oracle 10G XE to an SQL Server 2005 Database	200
Oracle 10G XE Server	200
Starting and Stopping the Oracle 10G XE Server	200
Using the Object Browser	201
Step 1: Creating a BI Project and Adding a Data Flow Task	203
Step 2: Adding an OLE DB Source and Configuring it to Connect to a Local Oracle 10G XE Server	204
Step 3: Configuring the OLE DB Source	205
Step 4: Adding a SQL Server Destination and Configuring its Connection Manager	207
Step 5: Establishing a Path from the OLE DB Source to the SQL Server Destination	209
Step 6: Configuring the SQL Server Destination Component	210
Step 7: Building and Executing the Package	212
Summary	212

Table of Contents

Chapter 14: Web Service Task to Convert Miles to Kilometres	213
Hands-On Exercise: Creating and Testing a Package that Uses a Web Service Task	214
Part One	214
Step 1: Create a Visual Studio 2005 Blank Solution	214
Step 2: Create a Web Service	215
Step 3: Create a WSDL File	217
Part Two	218
Step 1: Creating a BI project and Adding a Web Service Task	219
Step 2: Configuring the Web Service Task	220
Step 3: Building the BI project and Executing the Package	225
Summary	226
Chapter 15: Package that Transfers a Database from One SQL Server to Another	227
Hands-On Exercise: Creating and Testing a Package that Uses a Transfer Database Task	228
Step 1: Creating a Network Share on Computer 1	228
Step 2: Creating a BI Project and Adding a Transfer Database Task	229
Step 3: Configuring the Transfer Database Task	229
Establishing Connections	231
Configuring the Source Database Node	233
Configuring the Destination Database	235
Step 4: Building the BI Project and Executing the Package	236
Summary	238
Chapter 16: On Using Event Handlers in SSIS	239
Hands-On Exercise: Creating a Project with Two Packages	240
Step 1: Create a BI Project and Rename the Default Package	240
Step 2: Add and Configure the Package that has a Send Mail Task	241
Step 3: Add and Configure an Execute SQL Task to the Renamed Default Package	242
Step 4: Add an Execute Package Task to the OnError Event of the Renamed Default Package	244
Step 5: Add an Execute Process Task to the OnPostExecute Event of the Renamed Default Package	246
Step 6: Build and Execute the Package with Event Handlers and Verify	248
Summary	249
Chapter 17: Package that Transfers a File Using an FTP Task	251
Hands-On Exercise: Creating a Package with a FTP Task	252
Step 1: Creat a BI Project and Rename the Default Package	253
Step 2: Build and Execute the FTP Task	257

Table of Contents

Step 3: Disable the FTP Task, Add and Configure another FTP Task to Download a File from an Internet Public Site	257
Step 4: Build and Execute the New FTP Task	261
Summary	261
Chapter 18: Package with an ActiveX Script Task	263
Hands-On Exercise: Creating a Package with ActiveX Script Tasks	264
Example One: Word Automation	265
Step 1: Creating a BI Project and Adding an ActiveX Script Task to the Package	265
Example Two: Finding the Number of Tables in a Database	268
Step 2: Adding Another ActiveX Script Task to the Project	268
Example Three: Navigating to the Internet Explorer Browser	271
Step 3: Adding another ActiveX Script Task to the Project	271
Summary	272
Chapter 19: Package with a Script Task	273
Overview of the Hands-On Exercises	273
Hands-On Exercise: Creating a Package with Script Tasks	274
Simple Calculation	274
Calculation using variables	278
Add an Imports Statement to Build a String	280
Retrieve Data from a Database Table in the SQL Server 2005	282
Combine the Last Two Examples in Displaying Data and Copying to a File	284
Summary	286
Chapter 20: Package with Maintenance Plan Tasks	287
Backing Up a Database	289
Hands-On Exercise: Creating a Package with Maintenance Tasks	289
Summary	296
Index	297

Preface

SQL Server Integration Services, with the acronym SSIS, is a comprehensive ETL tool that made its debut with SQL Server 2005. It is a tool tightly integrated with Visual Studio 2005, having all the functionalities that its forerunner DTS (Data Transformation Services) had in SQL Server 2000. This does not mean that it is just an improvement over DTS, but a product that is totally different with a vastly improved interface; an extensible architecture; an enlarged tool set; ease of integration with other SQL Server Tools such as Analysis Services; capable of supporting connectivity with third party databases, and bringing into a central location many database management tasks.

Beginners Guide to SQL Server Integration Services Using Visual Studio 2005 provides you with the basic knowledge that you should have before you move onto more advanced ETL (Extraction, Transformation, and Loading). This step-by-step, hands-on guide will take you right into the Visual Studio 2005 Integrated Development Interface, making you appreciate and understand how Business Intelligence Projects and Packages are created using the Visual Studio designer. The book will also provide you with a comprehensive description of the many designer windows that you may encounter while working with the designer. This guide provides the building blocks, describing each block by way of an example, as well as describing the nuts and bolts that bind the blocks. You start building packages right from Chapter 2 and continue onto Chapter 20, gathering and building upon your knowledge in each step.

What This Book Covers

Chapter 1 will see the various roles played by SSIS in the enterprise business. You will also learn about the various objects used in SSIS, most importantly the Package object; the Control Flow Elements; the Data Flow Components; the Event Handlers, and other miscellaneous features of the VS 2005 IDE.

Chapter 2 will take a look at how to create your first business intelligence project for SSIS using the Visual Studio 2005 IDE. You will be introduced to the many windows that are used by SSIS specifically, and a few other windows of the VS 2005 IDE used for projects and solutions. The hands-on exercise at the end of the chapter helps you to examine on your own the key features of the IDE as well as how you may retrieve a VS 2005 created package using the SQL Server 2005 Management Studio.

Chapter 3 will cover how to send a mail by creating the Send Mail Task using the SMTP server that you can access. If you are not sure about your SMTP Server, you may find that Hands-On Exercise 2 gives you a better understanding of how to access the SMTP server.

Chapter 4 will take a look at how to create a SSIS package that transfers a table on your SQL Server 2005 to a flat file on your C:\ drive. You will also learn how to work with the Connection Managers that are so essential for data transformation tasks.

Chapter 5 will see how to create a SSIS package that transfers a table on your SQL Server 2005 database to an Excel spreadsheet. You will also learn how to connect to an Excel data destination.

Chapter 6 will create a package that transfers data from a table on SQL Server 2005 to a MS Access Database. You will learn how to work with the Data Viewer that monitors the data flow.

Chapter 7 will create a package that uses the Bulk Insert Task to transfer data in a text file on your computer to a table in the SQL Server 2005. You will also learn about creating a table on SQL Server 2005 using the Management Console.

Chapter 8 will create a package that conditionally splits the data from a SQL Server 2005 query and sends them to multiple destinations. You will also learn to work with Recordset Destination, an in-memory ADO Recordset Object, and use it to display the results of data splitting. In addition, you will learn about using Variables in a SSIS Package.

Chapter 9 will create a package that can aggregate data from a database using the Aggregate Data Transformation. You will extract data from a SQL Server 2005 and load it into a Recordset Object to review the aggregated results.

Chapter 10 will create a package that converts the data extracted from an Excel spread sheet source before loading it into a MS Access Database. You will also learn how to direct the data that is not accepted by the destination to another destination using the errors in data transformation.

Chapter 11 will create a package that shows you how to work with the XML Task. You will learn how to find differences between two XML files as well as applying XSLT (Transformation) to convert an XML file to an HTML file.

Chapter 12 will cover how to work with the various options of a File System Object and how to use a precedence constraint that orders the tasks before executing the Package. You will learn how to copy a file from one location on your computer to another.

Chapter 13 will take a look at how to copy a table on an Oracle 10G XE database to a database on the SQL Server 2005. You will also learn how to install an Oracle 10G XE server and work with its database objects.

Chapter 14 will create a package that returns the value returned by accessing a Web Service Task. You will also learn how to create a Web Service Task in Visual Studio 2005.

Chapter 15 will create a package that uses a Transfer Database Task from one SQL Server to another SQL Server (a different version). You will also learn how to access a SQL Server database on a network node.

Chapter 16 will create a package that uses event handlers and you will be learning about OnError and on OnPostExecute events. You will also learn about the ExecuteSQL Task as well as the Execute Process Task.

Chapter 17 will create a package for transferring files using the File Transfer Task. You will learn to work with both a local FTP site as well as a remote FTP site.

Chapter 18 will create a package using the ActiveX Script Task. You will have access to a fully commented code to help you along. You will be using this task to work with a word document; SQL Server Management Objects; and an internet browser object.

Chapter 19 will talk about the Script Task by creating packages that you use to interact with your file system as well SQL Server 2005 for a variety of tasks, such as making a simple calculation to retrieving data sets and loading into text files.

Chapter 20 will create a package that uses a Management Plan Task to backup a database on SQL Server 2005. You will also get a general understanding of the Management Plan tasks.

What You Need for This Book

This book assumes that Visual Studio 2005 Standard Edition has been installed on a computer running Windows XP with SP2. SSIS is integrated with SQL Server 2005 and hence it is assumed that SQL 2005 Server Standard Edition is installed.

Who is This Book for

This book is written for beginners in the developer track who are looking to get an exposure to SQL Server Integration Services; DBA's who are testing water with the Visual Studio IDE but without a wide programming experience; SQL 2000 Data Transformation Services users who are trying to move into SQL Server 2005 Integration Services; Microsoft programming professionals in Small Businesses who wear multiple hats (jack of all trades) – developer, programmer, and DBA with a little bit of experience in each of these.

It is expected that you know how to manipulate window objects, like clicking; dragging and dropping; using contextual help; tabbing, etc. It is also expected that you are exposed to SQL Server database basics and that you understand connecting to a database server; querying the database; reviewing objects; displaying and reviewing properties of objects, etc. Very little coding skill is assumed except that you can logically follow a fully commented code.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

There are three styles for code. Code words in text are shown as follows: "We can include other contexts through the use of the `include` directive."

A block of code will be set as follows:

```
<WebService (Namespace:="http://tempuri.org/")> _
<WebServiceBinding (ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
```

New terms and important words are introduced in a bold-type font. Words that you see on the screen, in menus or dialog boxes for example, appear in our text like this: "clicking the **Next** button moves you to the next screen".



Important notes appear in a box like this.



Tips and tricks appear like this.

Reader Feedback

Feedback from our readers is always welcome. Let us know what you think about this book, what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply drop an email to feedback@packtpub.com, making sure to mention the book title in the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or email suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer Support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Errata

Although we have taken every care to ensure the accuracy of our contents, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in text or code—we would be grateful if you would report this to us. By doing this you can save other readers from frustration, and help to improve subsequent versions of this book. If you find any errata, report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **Submit Errata** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata are added to the list of existing errata. The existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Questions

You can contact us at questions@packtpub.com if you are having a problem with some aspect of the book, and we will do our best to address it.

1

SSIS Basics

This chapter describes SSIS and its role in enterprise business. SSIS is governed by an object model, and has several important objects.

SQL Server Integration Services

Integrating data into applications or reports is one of the most important, expensive, and exacting activities in building enterprise data warehousing applications. SQL Server Integration Services with the acronym SSIS, is a high performance solution consisting of multiple tools dedicated to this task in MS SQL Server 2005 and is tightly integrated with the .NET Framework tools. SSIS is a product which has evolved over time from MS SQL Server 7.0, where it was called Data Transformation Service (DTS). Although DTS was the forerunner of SSIS, it is not just an improvement, but an entirely new product, which has all the features of the earlier tool, but with a vastly improved development interface, an extensible architecture, enlarged set of tools to address the varied and changing pattern of data in line with the latest developments in data related technology, looping structures, ease of integrating with Analysis Services, better management and performance.

ETL that is, extracting, transforming, and loading takes care of all the activities needed to accomplish data integration. This process consists of extracting data from a source, transforming the extracted data and then loading the modified data to a target server to be used by the application. There are a large number of independent vendors such as Ascential, Informatica, and so on, as well as several database vendors such as Oracle, IBM, and Microsoft in this market.

Enterprise data can be of very different kinds ranging from flat files to data stored in relational databases, with the more recent trend of data being stored in XML data stores. The extraordinary number of database related products, and their historic evolution, makes this task exacting.

It is not very frequent that extracted data is usable as it is, and may need some kind of transformation. Often, the extracted data has inconsistencies. There are many reasons for inconsistencies, as computer applications may be affected by changes in technology (version changes, new methodologies, etc.), poor or no validation, which are the norms in legacy data software changes (for example, the date and time will be new data types in SQL 2008 to address several issues of awareness of time zones, higher precision needed in financial applications, address compatibility with third party vendors, etc.), changes in the way applications interact with backend data, localization, etc.

Some inconsistencies can be understood quite easily, such as the same information stored in different data formats, while others could be more difficult.

The efficiency and agility of an enterprise would depend on the quality of data in its databases. Hence it is not only necessary to cleanse or scrub data, but also modify it to be internally consistent, and have the correct data quality. Low quality data has been one of the main causes of accidents and frauds in recent times. SSIS has built-in transformations such as fuzzy lookup and fuzzy grouping that can be used to clean, and standardize the data required by the target database. It is very important to standardize data across an enterprise, especially when data is originating from several subdivisions of the enterprise, where different sets of standards may be operative, or where data is consolidated from distributed systems.

Further, the target database may have a different data type from the one that is extracted, and in which case the data has to be modified. Data transformation or modification can range from very simple to very complex. While changing the data type of an extracted data to match the data type of the target database, or changing the case of the extracted data (string) can be simple, combining several columns based on some logic can be complex. What is nice about SSIS is that you can concatenate several simple tasks to arrive at a complex transformation with each link providing a specific change. As an alternative to concatenating transformations or tasks, using the script task with the full force of the .NET framework behind it, it can produce an optimized solution. This is indeed an awesome tool that should be put to good use.

When it comes to loading data to the target database, there is the challenging aspect of loading the data to match the metadata of the target. Many middleware programs, such as Siebel, have their own proprietary database on which they have built their application. In such cases, the incoming data has to be properly matched to the target database schema. Since the middleware uses data from different database products (MS SQL Server, Oracle, etc.), the ETL process has to step up to this task. In fact SSIS is best suited for this task, as a plethora of data access methods are provided. You will see an example of data going between SQL Server and an Oracle database in Chapter 13.

Objects Used in SSIS

SSIS is governed by an object model. Working with SSIS requires a clear understanding of this model. This section describes some important and salient features of the object model. The objects used in building an Integrated Services Package, and the manner in which they work together in the package workflow, are as follows:

- **Package:** This contains all objects in a single unit of work that can be retrieved, executed, or saved at one of several locations.
- **Control Flow:** These elements consist of tasks and containers, and take care of the work flow by preparing data, and arranging for interaction with other processes, but the ordering of tasks is controlled by precedence constraints, which is another important constituent of flow control.
- **Data Flow:** These components consist of a source (adapter) and destination (adapter) for the data, as well as any data transformation that may be needed. The source, destination, and the transformation stages are connected by a sequencing path to maintain the order of flow.
- **Connection Managers:** They stand apart, and facilitate connection to various sources for both extracting (sources) and loading (destination).
- **Variables:** They are used to communicate between various objects. The objects are isolated from one another by design to bestow security, manageability, and maintainability. The variables that support dynamic updating of column values, conditions used in establishing precedence of flow, and reference to record-set destination are some of the examples.
- **Event Handlers:** They run in response to events raised by other objects at run time. Every container has an event handler associated with it that executes at run time.
- **Log Providers:** These are objects, used for logging package run time information.

In addition to this, there are features that help in debugging and diagnosing problems in design. In the following sections, you will learn some of the basic details about each of the above topics. You will also be using them in the various exercises that you will be doing.

The SSIS Package

The package is an assembly consisting of the several objects described briefly in the previous section. It may also include other packages (nested package). Specifically it can consist of:

- Connections
- Control Flow Elements
- Data Flow Elements
- Event Handlers
- Variables
- Configurations

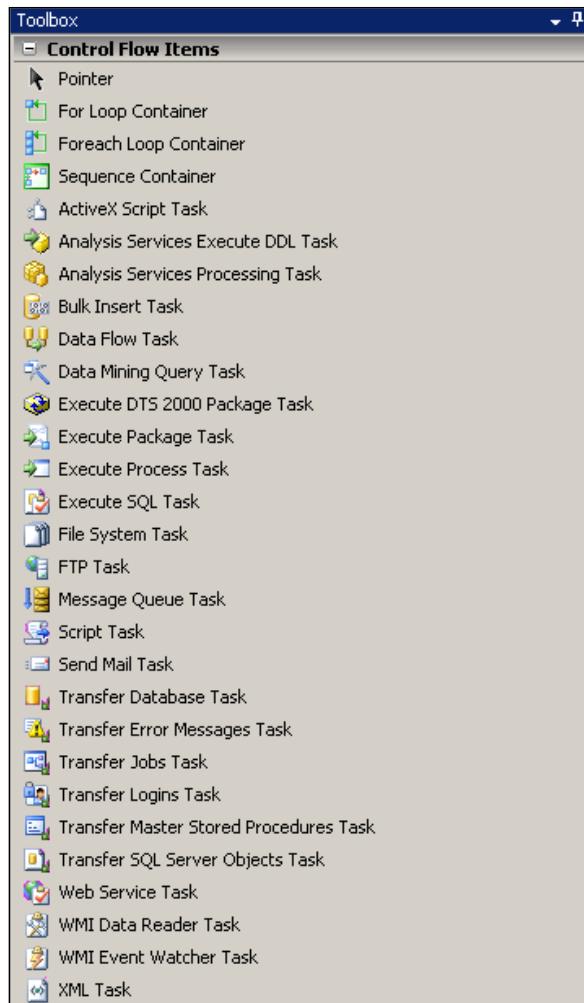
All elements are not necessary to build a package. In fact, a package can be empty; however, being empty, it will achieve nothing. In order to have functionality, it should have a control flow element, and one or more data flow elements.

A package can be either assembled using the graphical design tools that SSIS provides, or can be built by writing a program. The package built using these methods can be saved to Sql server, an SSIS Package store, or to a file system. The saved package can be retrieved, executed, modified, and saved.

The Control Flow Elements

As mentioned previously, the control flow elements consist of containers and tasks. The following figure shows the various items that you can find in the **Toolbox** that belongs to **Control Flow Items**. Tasks do all the work, and the containers provide the tasks with a medium to orchestrate them. Containers may contain other containers as well. The precedence constraints further orders the flow.

The list of tasks in SSIS includes most of the tasks that one would find in its earlier embodiment, the DTS designer. The various tasks can be more generically classified under—data preparation tasks, analytical services related tasks, scripting tasks, SQL Server tasks, maintenance tasks, etc. You will have the opportunity to work with some of these tasks in the various chapters of this book.



Of the available control flow tasks, the following are used in this book :

- Send Mail Task
- Data Flow Task
- Bulk Insert Task
- XML Task
- Web Service Task
- Transfer Database Task
- Execute Process Task
- File System Task
- Execute Package Task

- Execute SQL Task
- FTP Task
- ActiveX Script Task
- Script Task
- Maintenance Plan Task

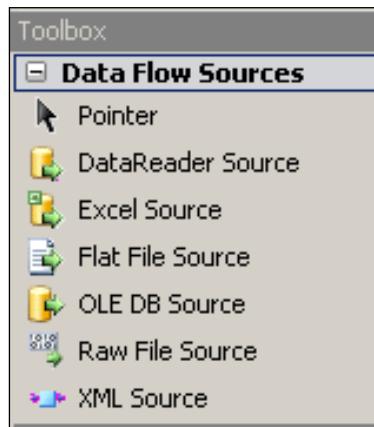
Data Flow Components

SSIS provides three types of data flow components – sources, destinations and transformations. Basically, data coming from the source is transformed before being loaded to the destination.

Data Source Components

Since extraction of data is one of the operations needed for ETL, SSIS is equipped with a rich interface for retrieving or merging data from heterogeneous data stores.

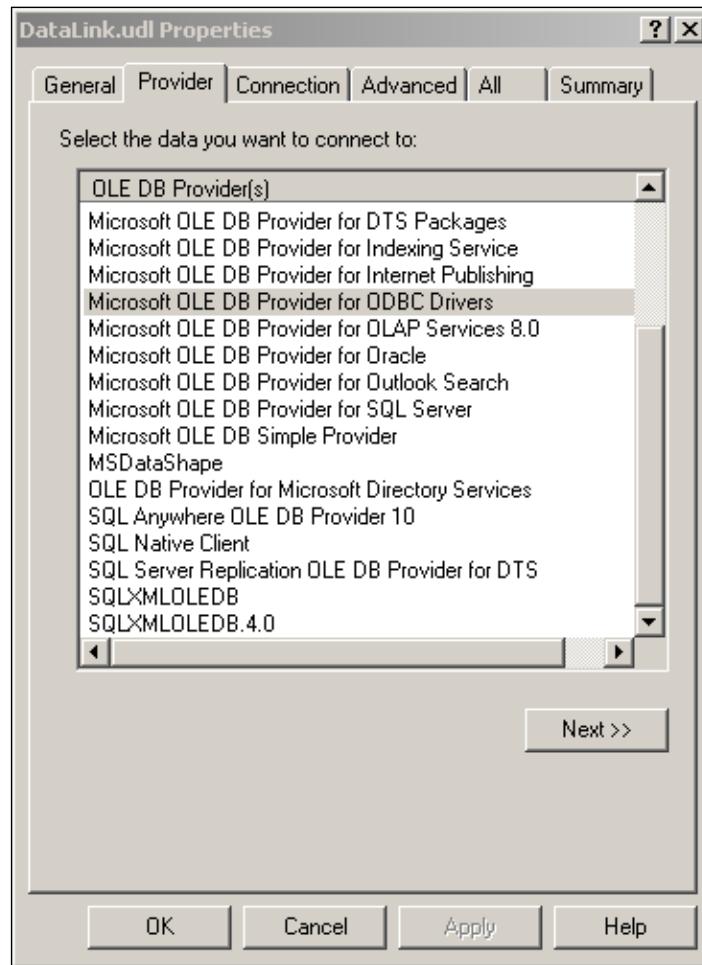
The variety of data sources that SSIS can connect to is rather large, using .NET, OLEDB, and ODBC data access methods. While .NET and OLEDB access methods are used for connecting to relational data, ODBC data access is used for legacy data. The ODBC data access method is especially useful, because there are ODBC drivers for practically every kind of database product. In addition to this, SSIS can also leverage data from flat files, XML files, MS Excel spread sheet files, and use MSOLAP provider, to access Analysis Services. The following figure shows various options for data source components available in the Visual Studio 2005 IDE (Standard Edition). It should be mentioned that the **DataReader Source**, a component of the .NET Data Access Strategy, connects to many of the vendors thorough the optimized .NET provider.



Of the available **Data Flow Sources**, the following are used in this book:

- **DataReader Source**
- **Excel Source**
- **OLE DB Source**

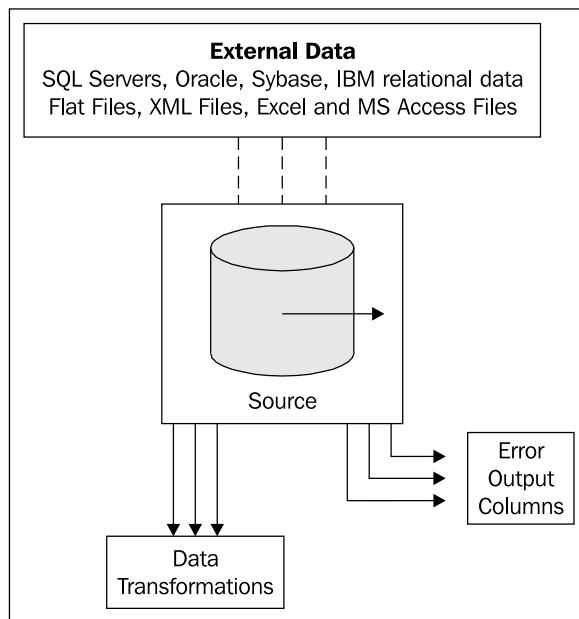
While .NET data providers were developed recently, OLEDB and ODBC have existed for a considerable amount of time and there exists a large number of both Microsoft and non-Microsoft products for establishing connectivity. The following figure shows a list, **DataLink.udl Properties**, which depends on OLEDB data access providers. It can be seen that there are a large number of providers, all of which can be used for connecting to data sources.



The following data access providers have been used in this book:

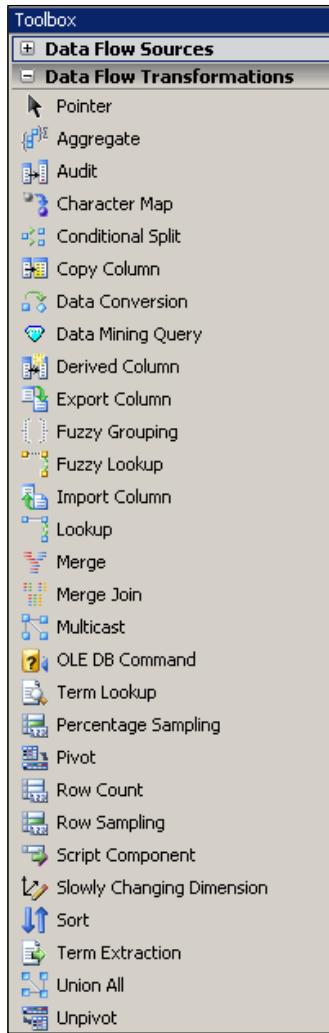
- .NET Provider\SqlClient Data Provider
- OLE DB Provider
- Microsoft Jet 4.0 OLE DB Provider
- Native OLE DB\Sql Native Client
- Native OLEDB\Microsoft OLE DB Provider for SQL Server
- Native OLEDB\Microsoft OLE DB Provider for Oracle

In SSIS, a source is typically a data flow component that conduits data from external data sources to other data flow components in a package. The following figure is a schematic representation of a **Source** obtaining its input from an external source and delivering it to the transformation stage. The error output has two columns describing errors in addition to the columns originating from the source.



Data Transformation

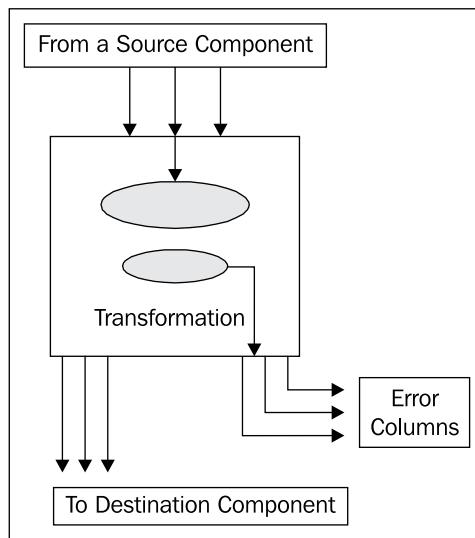
SSIS has built-in transformation that can be used to clean or scrub and standardize the data required by the target database. Standardizing data in an enterprise, especially when data is originating from multiple subdivisions having different standards, is important. Also, with globalization of enterprises, the importance of transformation has risen. The following figure shows the built-in data transformations available in Visual Studio 2005 (Standard Edition).



Of the available data transformations, the following are used:

- Conditional Split
- Aggregate
- Percentage Sampling
- Data Conversion
- Character Map

A transformation is an intermediary between a source and a destination. It receives input from the source, transforms it according to the type of transformation required and generates an output or outputs that will be written to the destination. It may also generate error output(s) that will be ported to other destination(s) as shown in the following schematic diagram. Although just one transformation is shown in the diagram, there could be a number of transformation stages (a chain of transformations) before the data reaches the destination.



A brief explanation of what each of these data flow transformations contribute to package design is given in the following sections:

Aggregate

Aggregate functions used in the SQL syntax such as SUM, AVERAGE, etc. are provided by this transformation. This transformation can either provide a value, or a warning that can be used in another process. The processing is carried out asynchronously on the data. It can also be used in formulating SQL syntax using the GROUP BY clause. will find an example in Chapter 9.

Audit

This transformation supports inclusion of environment information into the package in which it is used. This enables auditing the package. This has only one input and one output. Audit information is added to the data flow information.

Character Map

This transformation takes care of string transformations. A string in a column can be replaced, or a new column can be added with replaced strings. For example, first name, last name, etc. with capitalized first letter can be generated, while the input comes in with arbitrary cases. You will find an example in Chapter 5.

Conditional Split

Routing the data conditionally to outputs, based on a given criteria is sometimes needed. For example it is possible to sort out a list and fill items A to M in one table and N to Z in another table, etc. You will find an example in Chapter 8, based on this.

Copy Column

This transformation copies a column to a new column to which you may attach other transformations such as aggregate, character map, etc.

Data Conversion

Data type mismatch is not an uncommon source of error during transformation while extracting from data sources, or loading data to target databases. Data truncation errors in strings as well as numbers are possible. Error outputs can be used to divert mismatched items to a different location, which can be looked at more closely and modified. You will find an example in Chapter 10.

Data Mining Query

This transformation is really a task that is used in conjunction with Analysis Services. DMX or Data Mining Extensions language is used to obtain results in the form of a single data value, or a data set, by running a query against Analysis Services data by connecting to Analysis Services or to an Analysis Services project.

Derived Column

As the name implies, you can create a new column by combining columns, or by applying mathematical expressions to values in columns, or populating a column with a part of data in a column. For example, you multiply price and quantity column values to produce a total price column.

Export Column

Data read during data flow can be directed to a file. Product information components such as a graphics, other product information, or a drawing, can be directed to a file where it gets stored. For this transformation, only specific SSIS data types are used, such as—DT_TEXT, DT_NTEXT, or DT_IMAGE, as it is specifically used to divert images and documents.

Fuzzy Grouping

This transformation is mostly used in data cleansing, by separating duplicate data and assigning a marker to show how data is duplicated with reference to a chosen row (called a canonical row). Departures from the exact match are scored, but it does not remove duplicates. For each input row in data, three additional columns are produced after processing, which are: indicating the location of the row, identifying a group of rows which may contain duplicates, and score showing the goodness of match.

Fuzzy Lookup

This transformation, unlike lookup transformation, uses fuzzy matching. It is not used for locating exact matches, but for locating close matches in the data flow. Fuzzy matching is carried out against a reference data source in SQL Server table. Again, there are restrictions on data type for using this transformation being limited to `DT_WSTR`, and `DT_STR`.

Import Column

This transformation reads data from files and adds it to database columns. It can bring in data from files together with associated resources such as graphics, binary documents, etc. into database columns. Again there are restrictions on the output data type being limited to `DT_Text`, `DT_NText`, and `DT_Image`.

Lookup

This transformation looks for data in a reference database. The package that uses this transformation requires a dataflow task. It is carried out via `EQUIJOINS` with the reference database.

Merge

Merge transformation is similar to "union all" transformation. It simply merges two sorted datasets into a single dataset. Merging is carried out using key column values. Merging data from two data sources, such as tables and files, nested merge transformation, etc. are possible.

Merge Join

Merge join by definition, requires at least one data flow task and two data flow components to participate in the join. Just like merge transformation, merge join transformation also requires sorted inputs. Several different kinds of joins, such as `LEFT JOIN` or `LEFT OUTER JOIN`, `RIGHT JOIN` or `RIGHT OUTER JOIN`, `FULL JOIN` or `FULL OUTER JOIN`, and `CROSS JOINS` are possible using this transformation.

Multicast

As the name implies, the data is copied to multiple outputs (every row is copied). This is different from a conditional split; it can be used in cases where you may want to apply different transformations to the same data set getting sorted, while the other undergoing some kind of aggregation.

OLEDB Command

Parameterized queries are a powerful means to extract just the information needed to satisfy a given criteria. You would use a parametric query when you try finding patient information, given the patient-ID from a patient database. This transformation requires a data flow task, and an OLEDB data source or a flat file source.

Term Lookup

This transformation and term extraction transformation that follows provide support for word search in text documents. The frequency of occurrence of words is captured using a reference containing words to be matched. This is an excellent tool for preparing custom lists of words in a text with statistics.

Percentage Sampling

This transformation allows you to sample out a percentage of the data (Percentage of total rows) which may be useful in testing a package with a smaller set of data. For example, instead of using the full thousand rows of data, the package may be tested with a smaller number of the rows, which is representative of the entire set. You will find an example in Chapter 9.

Pivot

While normalized data are a standard for OLTP applications, de-normalized data are common in OLAP. A list of rows containing data does not reveal hidden information in a simple listing, but by carefully grouping data, hidden features can be revealed. Pivot transformation does it automatically if it is properly configured.

Row Count

As the name implies this transformation counts the number of rows that go through the data flow. The row count is stored in a variable associated with this transformation.

Row Sampling

This is very similar to percentage sampling transformation. The difference is that you can choose the size of the dataset and you will get a randomly selected subset of data. When you need to randomly choose a few rows from a large group, this is ideal. Just as in the case of percentage sampling, the seed for random selection can be chosen, or you can use a machine generated seed.

Script Component

If you need data flow programming using all the power of .NET, then this is the transformation to use. You can add script support for data flow to work at run time using the powerful programming constructs in Visual Basic, complied and executed at run time.

Slowly Changing Dimension

This transformation is used with both OLTP and OLAP to attend to changes that take place in dimension tables or look up tables. When an attribute of a row changes in either of these, there are situations when you want to keep the old value as well as the new value. This transformation has been designed to address several kinds of business needs, also known as change types. Some of the change types are: Changing Attributes Updates Output (basically an overwriting of an existing record); Historical Attribute (creating a new records instead of updating the records); and Fixed Attribute (Keeping the value unchanged).

Sort

When you need columns to be sorted either in ascending, or in descending order, this transformation can be used. The input list can be sorted in multiple ways, starting from the lowest column sort order. As the data to be sorted may contain strings, a string of comparison options can be defined for this transformation. Unsorted columns have a sort order of 0.

Term Extraction

This transformation is basically used with text data of type DT_WSTR or DT_NTEXT. This transformation extracts nouns only, noun phrases only, or noun and noun phrases. It cannot extract articles, verbs, adjectives, pronouns, etc. and it writes the extracted items to a transformation output column. The transformation works only for English language and uses an internal dictionary to aid in the extraction. The transformation normalizes the case of the text extracted by default to lower case.

Union All

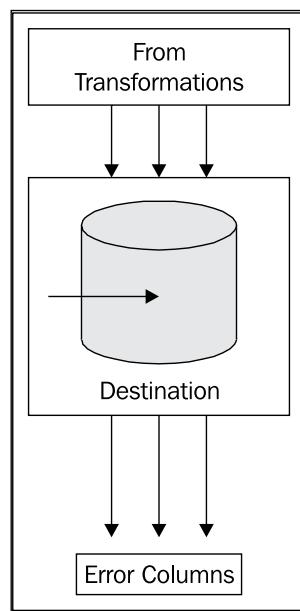
This transformation is similar to an SQL statement with a UNION ALL clause. You can combine the values of two sources to create a combined source containing inputs in the sense of an SQL statement. The two sources should have the same metadata for this to work.

Unpivot

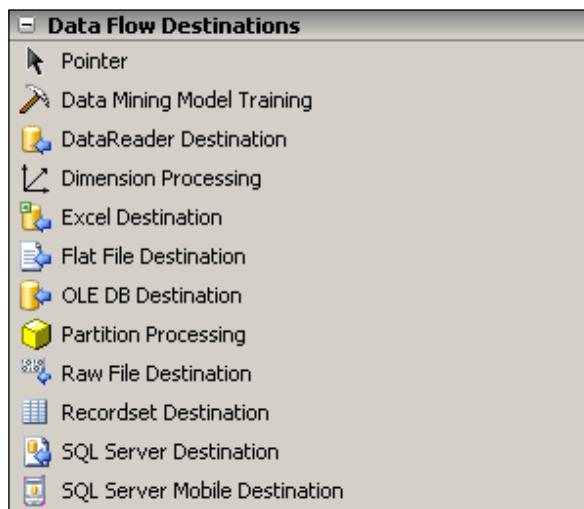
This transformation is the reverse of the pivot transformation mentioned earlier.

Data Flow Destinations

Similar to the data flow sources, the SSIS designer provides a number of data flow destinations. The data is written to the destination data store or to an in-memory data set. The destination must have a minimum of one input. As will be seen, the input may be a column, or columns from a source table after undergoing some kind of transformation. In addition to an input, the destination may also have an error output. This is designed for the eventuality that if some data cannot get into the destination due to some constraints, it can be routed to the error output. The following schematic shows the data flow destination with multiple inputs and error outputs. SSIS neither has restrictions on the number of inputs nor on the number of error outputs.



The following screenshot shows the various destination options available in the SSIS designer toolbox. The OLEDB destination in turn supports connection to various database vendor products, for which OLEDB providers are available.



Of the available **Data Flow Destinations**, the ones that are described in the book as follows:

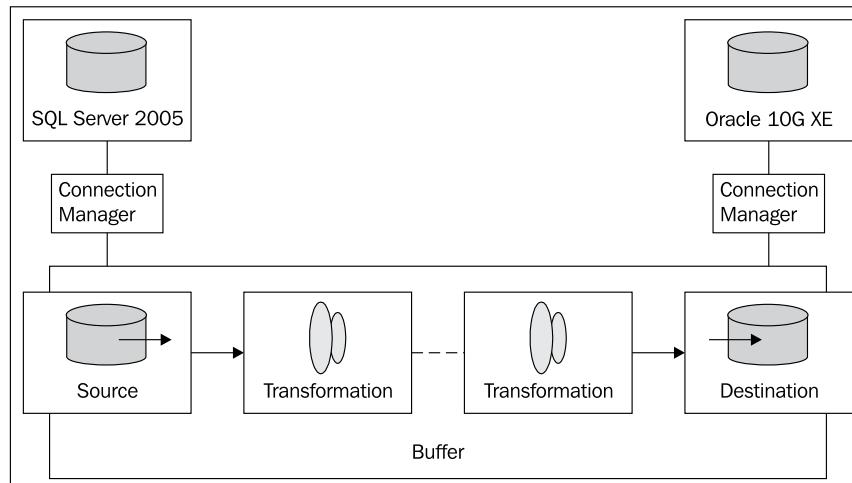
- **Excel Destination**
- **Flat File Destination**
- **OLE DB Destination**
- **Recordset Destination**
- **SQL Server Destination**

Connection Managers

For the source to obtain its input, it has to use a connection manager that configures the connection to the required data source. Similarly, while loading to a destination, the connection manager manages the connection to the external source. The connection managers arrange for valid connection to the data sources. Although the bulk of activity in SSIS is database related, some of them are not, for example, FTP task or Send Mail task.

A connection manager represents a logical connection to data. By using the `ConnectionString` property, set at design time, it establishes a physical connection to the data source at execution or run time. The `ConnectionString` property would depend on the type of data source.

The following figure shows a schematic diagram of how the connection managers work with the source and destination adapters in a data flow task. Connection manager is a device that points to where the data is located (the servers, the files, etc). Different vendors have different ways of indicating this information. In the figure shown, the **Connection Manager** connected to the source is pointing to an **SQL Server 2005** and the destination is pointing to an **Oracle 10G XE** database server. The data flow transformations take input from the upstream side (coming from the **Source**) to the next data transformation stage if it exists. There may be many data transformation steps. Finally, the data is passed to the destination adapter which will write to the **Destination** after conferring with the connection manager at its end. It is possible that the servers may change or move, all that will be required in such instances is to reconfigure the connection.



In order to cater to a variety of data sources, there are different types of connection managers. The following table shows a partial listing of the connection managers:

Manager	Description
ADO Connection Manager	Connects to data source using ActiveX data objects method for data access.
ADO.NET Connection Manager	Connects to data source using the .NET provider.
Excel Connection Manager	Connects to MS Excel Spread Sheet data.
File Connection Manager	Connects to files and folders.
Flat file Connection Manager	Connects to data in a single flat file.
FTP Connection Manager	Connects to a FTP Server.
HTTP Connection Manager	Connects to a HTTP or Web Server.

Manager	Description
OLE DB Connection Manager	Connects using a OLEDB Provider.
SMTP Connection Manager	Connects to an SMTP Server.
ODBC Connection Manager	Connects to a data source using ODBC.
SMO Connection manager	Connects to SQL Management Server Objects.

Variables

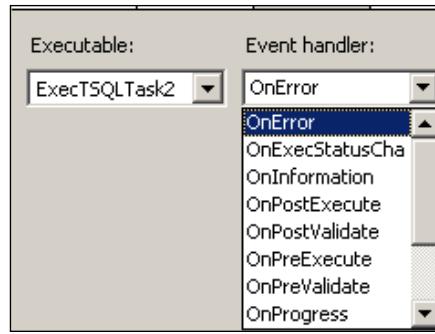
Variables are used to store values that package objects can use at run time. They assist in communication between objects. Event handlers, containers, tasks, etc. can all make use of variables. Variables can also be used in the scripting tasks. Precedence constraints that sequence tasks can also make use of variables in the definition of constraints. You will see a number of examples of using variables in the book.

User-defined variables are created by package developer, and system variables are defined by Integration Services. Both user-defined variables as well as system variables are case sensitive. While user-defined variables can be defined as and when needed, system variables are fixed.

A variable is known by its name, description, namespace, read-only or read-write attribute, whether or not it raises an event, and data type and value. It is created in the scope of the package or a package object.

Event Handlers

Events of various types occur one after another from the time you start to execute a package till the time it completes successfully or otherwise. The **OnError** event, which is raised when an error occurs, is the default event in SSIS. In addition to the package, various components such as for loop, task host containers, etc. can raise events. Events can be raised by other executables such as for loop, and task host containers. You can create a custom event handler that traps these errors for some useful purpose. For example, you could send an email when a task fails. Besides **OnError**, there are many other types of events raised by the executables as shown in the following figure where the executable package is called **ExecTSQLTask2** (file name of package, **ExecTSQLTask2.dtsx**).



Containers in a package follow a hierarchy. If the event in a container is not handled, it can escalate the handling of the event to a container higher up in the hierarchy, quite like events on a web page which also have a container hierarchy. If there are no event handlers configured at any of the containers, the event never gets handled.

Like other objects in SSIS, event handlers need a name and a description. You may need to indicate whether or not the event handler runs and also indicate whether the package fails if the event handler fails; you may also indicate an execution result to return instead of the actual execution result for the event handler at run time. Also, an event handler can have a transaction option as well as a logging mode specified. You will see an example with two event handlers in a package in Chapter 16.

Log Providers

Logging is an important concept, especially when it comes to auditing and troubleshooting problems that occur at run time.

SSIS has built-in log providers that can be used to implement logging in packages, tasks, containers, etc. Logging can capture run time information about a package, such as the user who ran the package and the start time and stop time of the package execution. Adding logging support to a package requires a log provider as well as a location for the log. A log provider specifies the format for the logged data. Following are the types of log providers:

- Text file log provider, log will be in ASCII format in a CSV file.
- SQL Server Profiler log provider, provides a log file with extension .trc.
- SQL Server log provider writes entries to sysdts1g90 table in SQL 2005 database.
- Windows Event log provider, events log in the local computer.
- XML file log provider, generates a log file with extension .xml.

Debugging and Diagnostic Features

It is possible that a package has a bug, and in order to look into where the problem is, you need debugging features. SSIS has built-in debugging support in the form of **Breakpoints** window, **Locals** Window, **Watch** window, and the **Call Stack** window. These function similar to what one finds in the other Microsoft product like Visual Basic. Additionally, there are diagnostic windows like **Error List** window, **Output** window, and **Task List** windows. While executing a package, you must make use of the **Data Viewer** diagnostic tool. You will be using data viewers in Chapters 6, 8, 9, and 10.

Summary

In this chapter, we described the role played by SSIS in improving the productivity of enterprise business. The arsenal of tools available for the package designer using the VS 2005 IDE is described. As a major portion of the tasks are related to data, the objects participating in the data flow are described in greater detail.

2

Creating a BI Project for SSIS in Visual Studio 2005

The previous chapter described several key objects in the SSIS toolbox. This chapter completes the picture by describing how a business intelligence project for SSIS is created using Visual Studio 2005 IDE. It is important to know how such a project is created, and understand the details of the various menus, objects, and windows that are displayed, and their role in package design. Although somewhat overwhelming, you will get used to the various details as you work through the chapters. Hands-on exercises should help you in creating, renaming, saving, and importing packages.

Business Intelligence using Microsoft Products

Microsoft provides an integrated business intelligence suite of applications that offers enterprises end-to-end support, starting with its three-piece suite – Integration Services, Analysis Services, and Reporting Services. Excel applications, Share Point server, and Office 12 suite gives it a further boost by tight integration. It is in SSIS that the integration of enterprise business data takes place where raw data is extracted, cleansed, transformed, and loaded back to the server to be analyzed in a complex and interactive manner before being disseminated in intelligent and interactive ways to enable decision makers with the right kind of data, at the right time, in a format easily understandable by them.

Resources Used for Creating Projects

In this chapter, we begin with creating a business intelligence project with SSIS using Visual Studio 2005. In order to create such a project, it is first of all necessary to have Visual Studio 2005 software installed on your computer. In addition, as data on servers are accessed by the project, the computer should have access to database servers such as SQL Server 2005, Oracle 10G XE server, SQL Anywhere 10 server, etc., as well as other sources of data such as Microsoft Access Databases, Excel Spread Sheet data, XML files, etc. If files and folder structures on the system are accessed, then permissions to use them must be in place. For working with Analysis Services, the analysis server should be accessible. It is assumed that some or all of these resources are available.

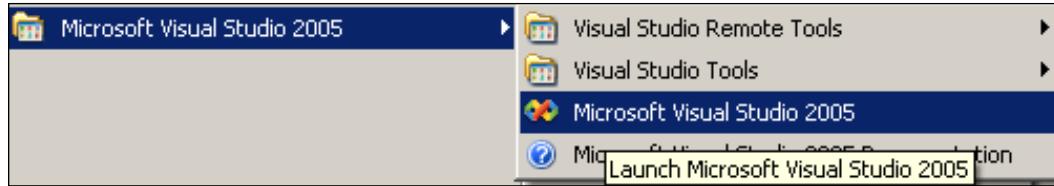
For this chapter it is assumed that Visual Studio 2005 Standard Edition has been installed on a computer running Windows XP with SP2. SSIS is integrated with SQL Server 2005, hence it is assumed that SQL 2005 Server Standard Edition is installed. SSIS is not supported in SQL Express (<http://technet.microsoft.com/en-us/library/ms165636.aspx>).

Creating Your First BI Project for SSIS

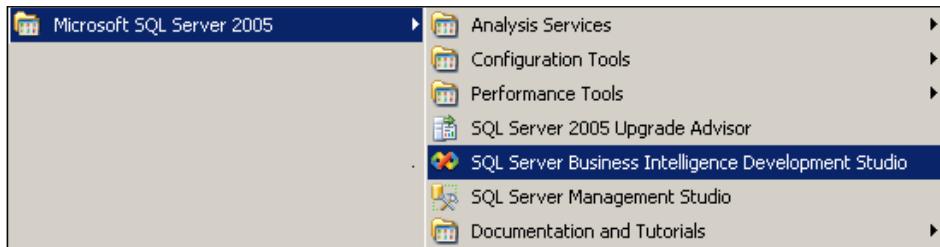
In this exercise, you will create a BI project for SSIS with a default package that is present, and then review the Microsoft Visual Studio IDE to get acquainted with the various windows that are opened; some of which are specific to SSIS. You will also review the properties of the default package created with the project.

Launching VS 2005 and Creating a BI Project for Integration Services

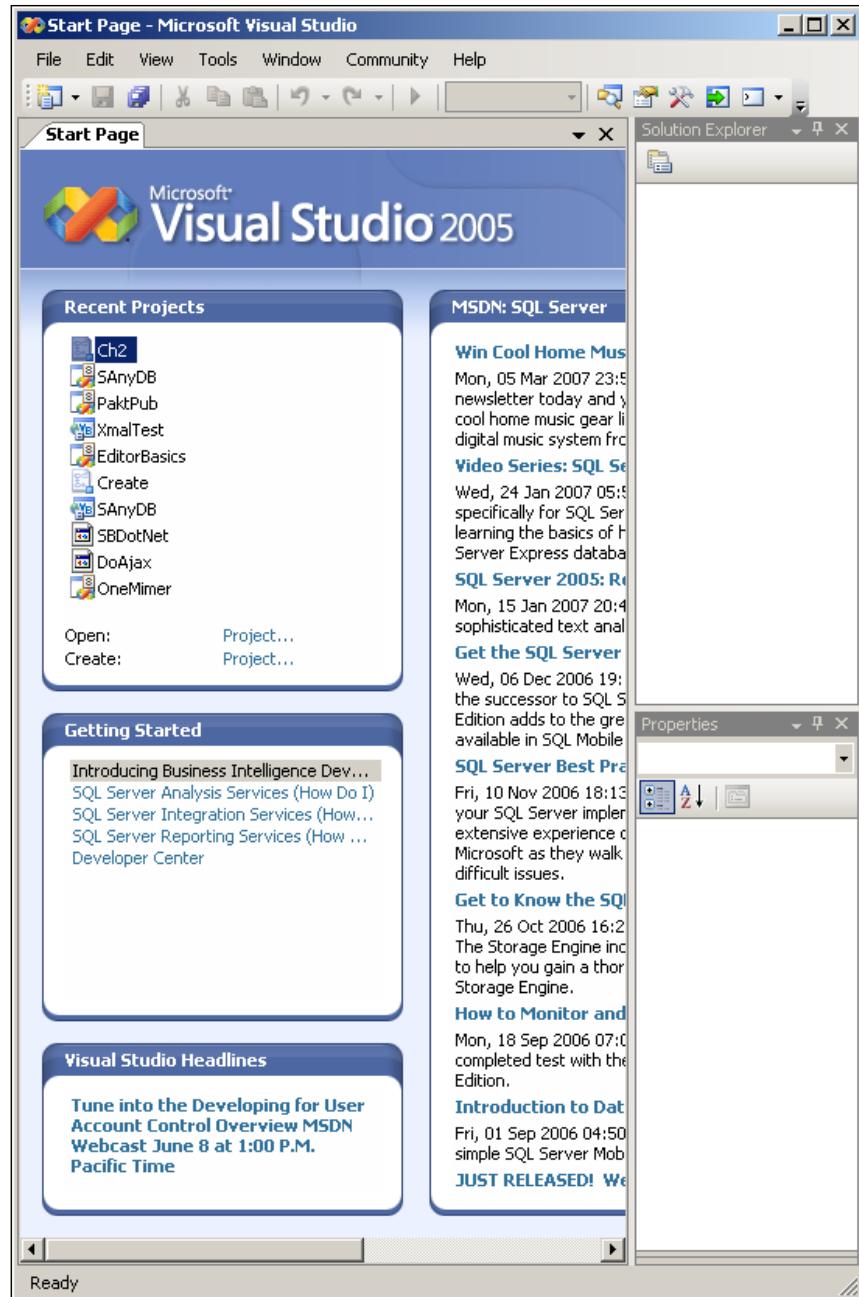
When Visual Studio 2005 is installed, a shortcut will be added to the **All Programs**, which is visible when you click on the **Start** button of your computer. Now launch the program by clicking on the drop-down menu item, **Microsoft Visual Studio 2005**.



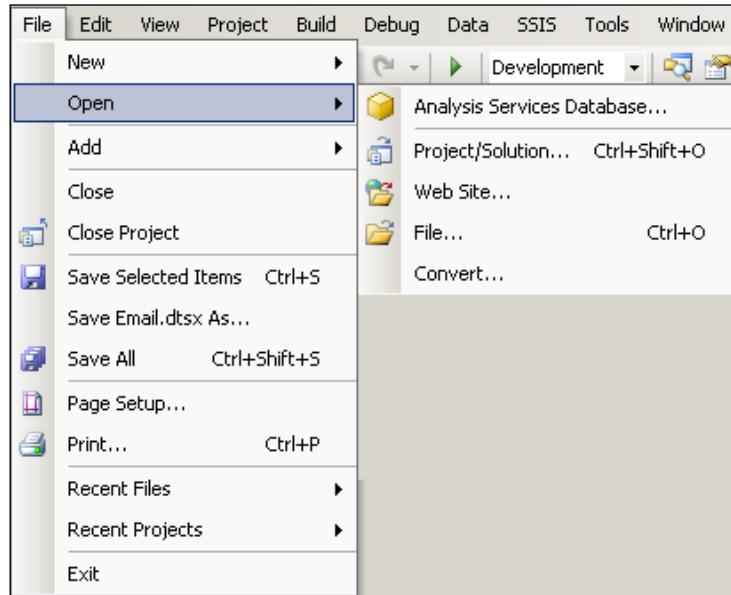
You can also launch the same from a menu from the **Microsoft SQL Server 2005** shortcut, as shown in the following screenshot.



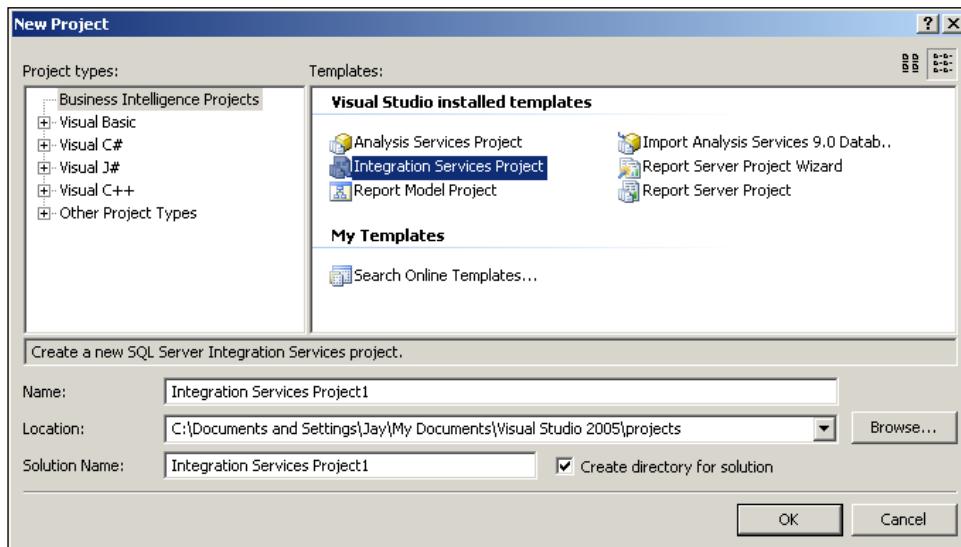
Microsoft Visual Studio IDE application is launched, showing the **Start Page** (as shown in the next screenshot), listing all the installed programs and add-ins. This window also shows you projects that you have worked on recently under **Recent Projects**. It has an empty **Solution Explorer** window and a bare **Properties** window. **Solution Explorer** is a container to which you can add projects that you want to create or modify. The links to the various **MSDN: SQL Server** related content on the internet is provided by the window next to the **Recent Projects** window. This window gets updated and you will be able to access new information. The **Getting Started** window gives you links to your learning experience (how do I do this in BI?) immediately available. Related content and resources may be accessed using the **Community** menu item, as this provides a gateway to the MSDN forums.



When you click on the **File** menu item, it will display a drop-down menu with several items. Using the sub-menu item **Open**, you can access a number of items as shown in the next screenshot, including access to **Analysis Services Database....**



Using the sub-menu item **Open**, you can create a project, a website, or a file. Click on **File | New | Project** to create a new project. This opens the **New Project** window as shown in the following screenshot:

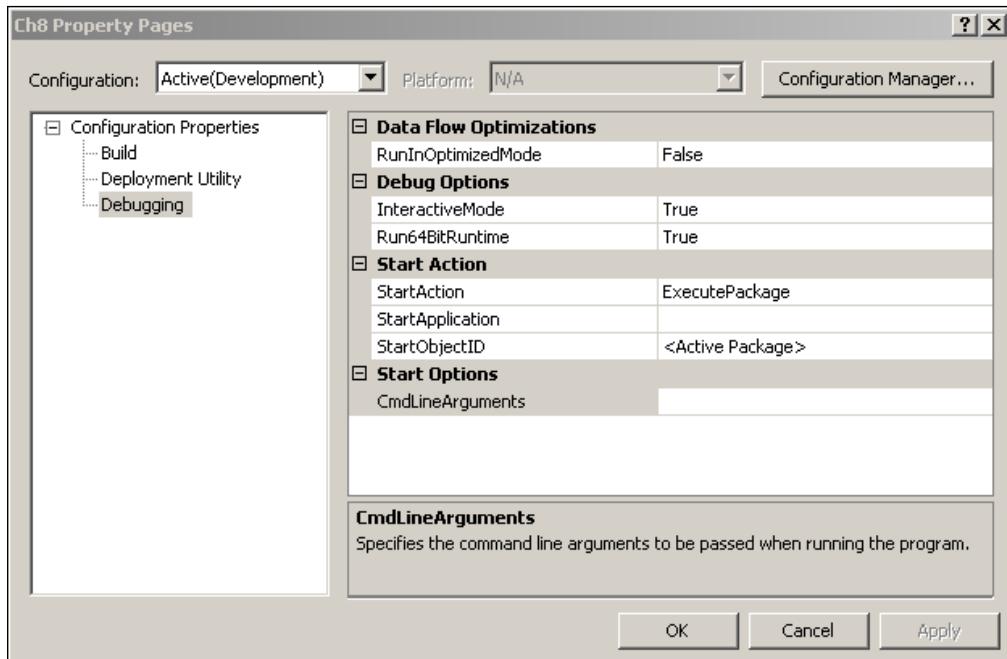


There are several project types that you can choose from, including **Business Intelligence Projects**. This group consists of the three main business intelligence projects: **Integration Services Project**, **Report Model Project**, and **Analysis Services Project**. Click on **Integration Services Project** in the **Visual Studio installed templates** group. This will display **Integration Services Project1** in the **Name** field on this window as shown. This file will be located at a default location, which is **C:\Documents and Settings\<user_id>\My Documents\Visual Studio Projects**. At this point, you may provide an appropriate name for this project such as, **Ch2**. In the **Name** field, replace **Integration Services Project1** by **Ch2**. The project can also be saved to a different location by browsing to it using the **Browse** button if required. When you click on the **OK** button after providing a name, you will have created the project.

Business Intelligence Project Properties

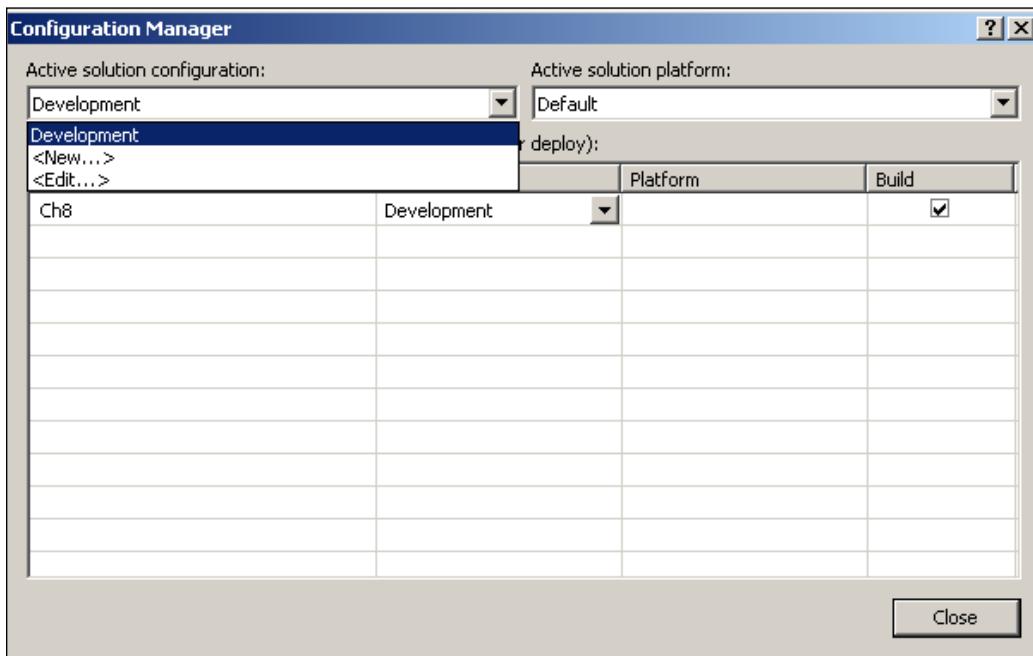
The **Business Intelligence Project Properties** window can be accessed either by right-clicking on **Project** in **Solution Explorer** and choosing properties from the drop-down menu, or by going to **Project | Properties** menu.

The following screenshot shows a typical window that pops up when you choose to display the properties of a project. The screenshot shows the default **Debugging** option of the project.



Build, just displays the output path indicating the location of the output files generated during the build process, which is by default in the **bin** directory of the project. **Deployment Utility** displays by default the **Deployment Output Path**, whether or not you want to create a deployment utility, and whether or not you allow changes to be made to the configuration.

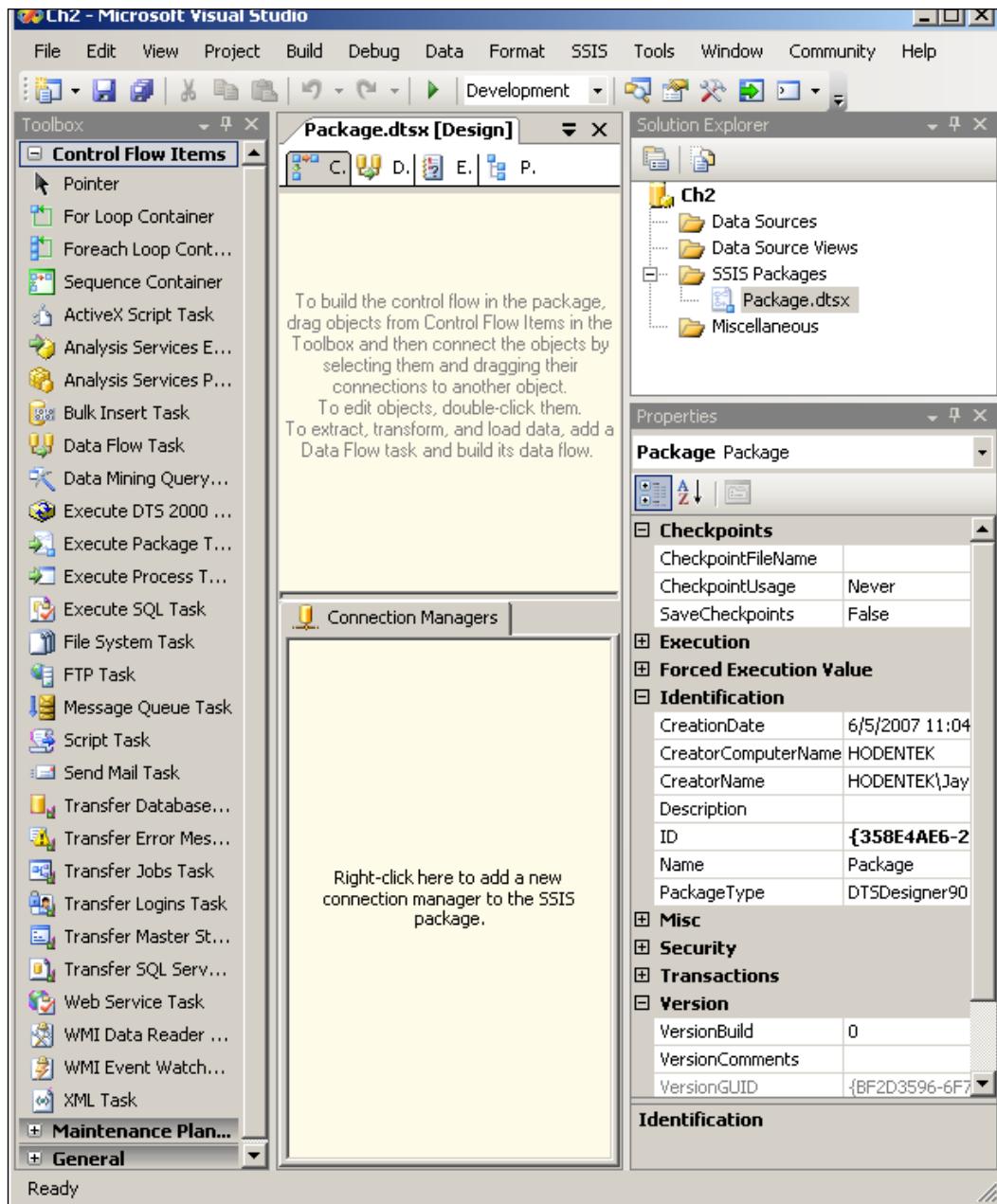
Clicking on the **Configuration Manager...** button would give a window showing the panel to administer the configuration as shown. **Active solution configuration:** as well as **Active solution platform:** can be modified in this window.



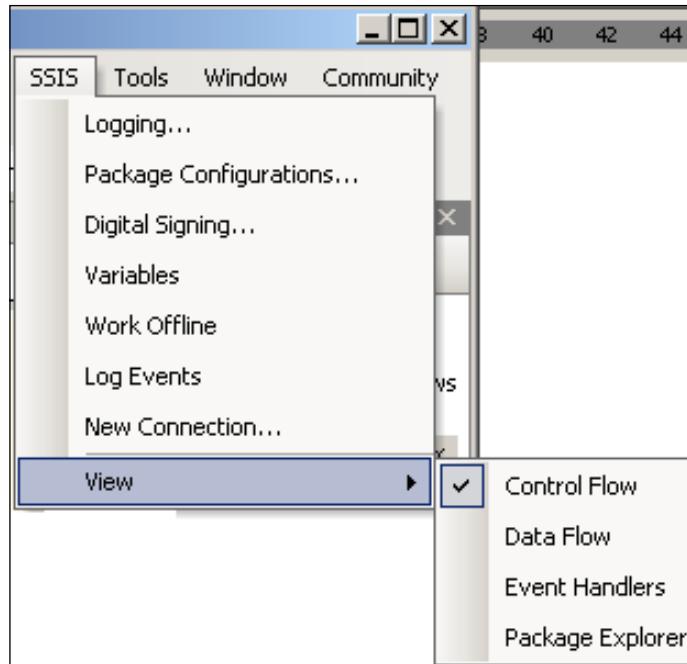
Overview of the Project Window

When the **OK** button is clicked, a skeleton project with a default package named **Package.dtsx** and three other folders, **Data Sources**, **Data Source Views**, and **Miscellaneous** are added to the **Solution Explorer** window, as shown in the next screenshot. The default package is empty when created with the name, **Package**. It can be renamed to a chosen name, usually descriptive of its intended purpose. During package design, you can drag and drop appropriate objects from **Control Flow Items**, **Maintenance Plan Items**, and **Data Flow Items** (Data Source, Data Flow Transforms, and Data Destination) windows in the toolbox. The design window of the package is shown wedged in **Toolbox** and **Solution Explorer** in the next screenshot. It is in this window that most of the design is carried out.

Creating a BI Project for SSIS in Visual Studio 2005

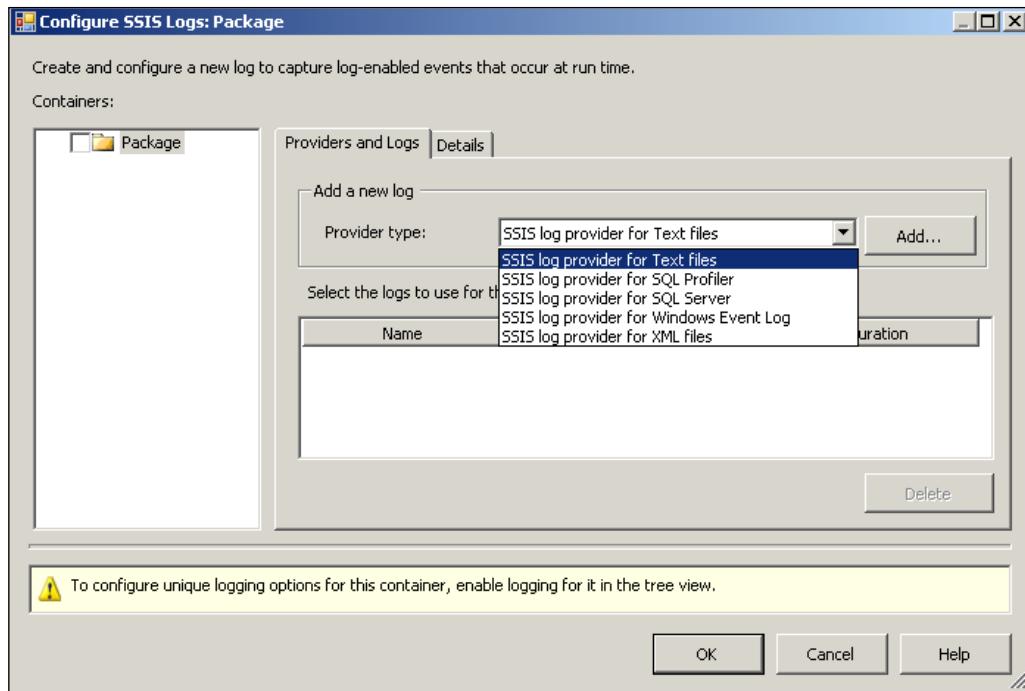


You may also notice a new menu item, **SSIS**, added to the main menu. The sub-menu items of this menu are contextual. This menu item has the following sub-menu, and you get it to display as shown, provided you first clicked on the **Package Designer** pane.



The same menu can also be displayed with some additional items by right-clicking on an empty area in the Package Designer's **Control Flow** tabbed page. This is also contextual and will be displayed when you click on the **Control Flow** tabbed page first.

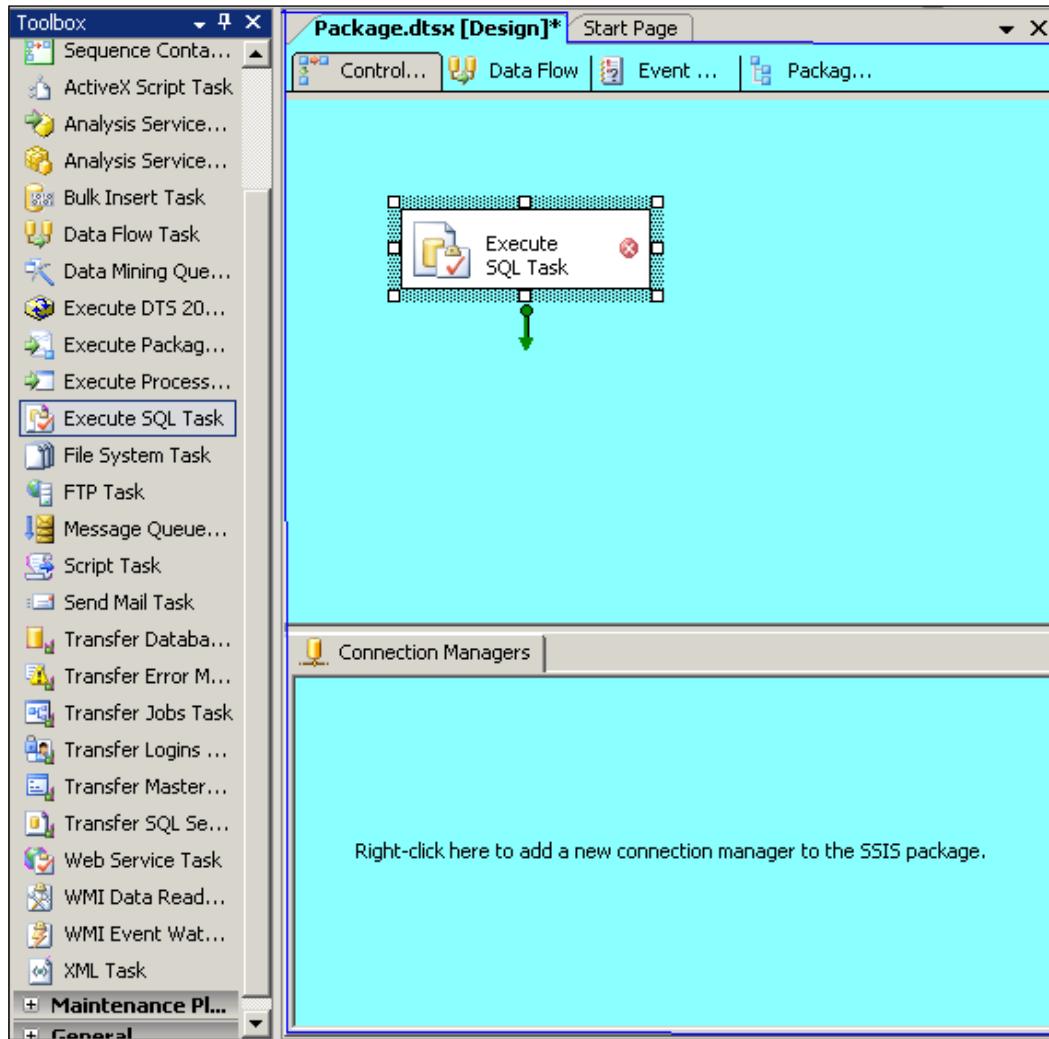
These sub-menu items are important in the package design. For example when you click on the sub-menu item, **Logging...**, you open up the window shown in the next screenshot that allows you to add a log provider to your package. You will see all the log providers discussed in Chapter 1 here.



Just below the **Solution Explorer** window, you will find the **Properties** window, where a wealth of information regarding the package is displayed. The **Identification** property is shown expanded in the window. The package name, the creator name, the created date, etc. are all documented here. Expand other properties and get acquainted with the information. Security is obviously one thing you may want to look up and, if needed, set up a password for the package.

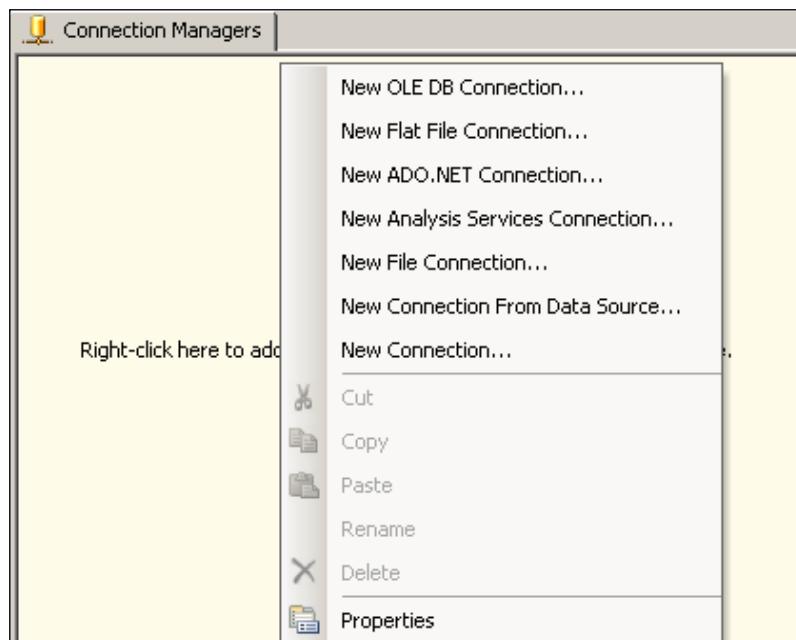
Canvas for Package Design

The Canvas for package design is shown wedged inbetween the **Toolbox** and the **Solution Explorer**, as shown in an earlier screenshot. The following screenshot shows this Canvas, consisting of four tabbed pages at the top – **Control Flow**, **Data Flow**, **Event Handlers**, **Package Explorer**, and the **Connection Managers** tab at the bottom.



Control Flow

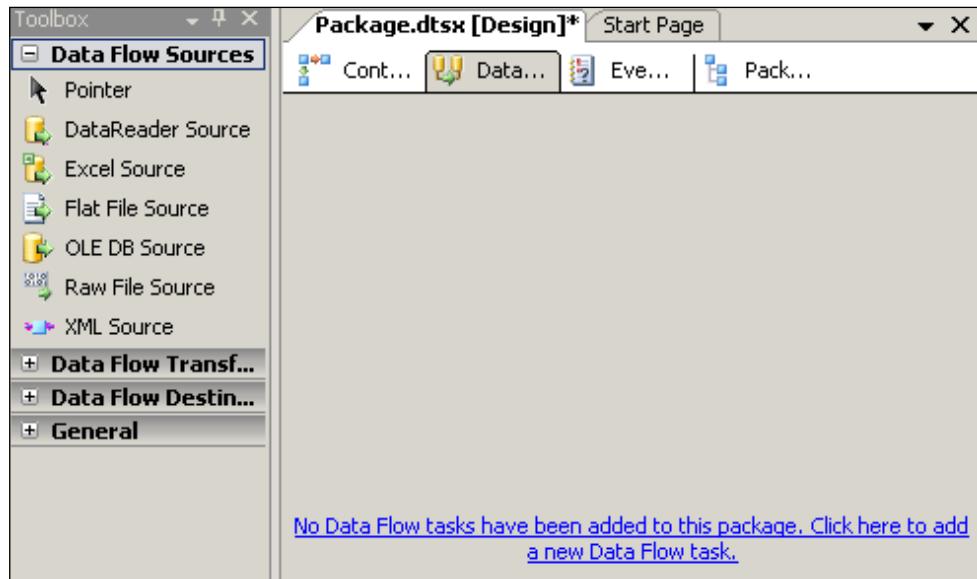
Control Flow items group will be displayed when you choose the **Control Flow** tabbed page. You can drag items from **Control Flow**, as well as **Maintenance Plan Tasks** toolbox groups to this page. For example an **Execute SQL Task** has been dragged to the **Control Flow** page in the previous screenshot. (You can double-click on an item in the toolbox and add the item to **Control Flow** also.) This object has a red circle with a white cross. This indicates that the object's configuration is incomplete. Directly below the **Control Flow** tab on the same page is the tabbed page (tray), **Connection Managers**. The connection manager used by the control flow tasks will be placed here. Again the instruction is clear, to add a connection manager, right-click and add by choosing the correct drop-down menu item. When you right-click in the tray, you will display the following menu:



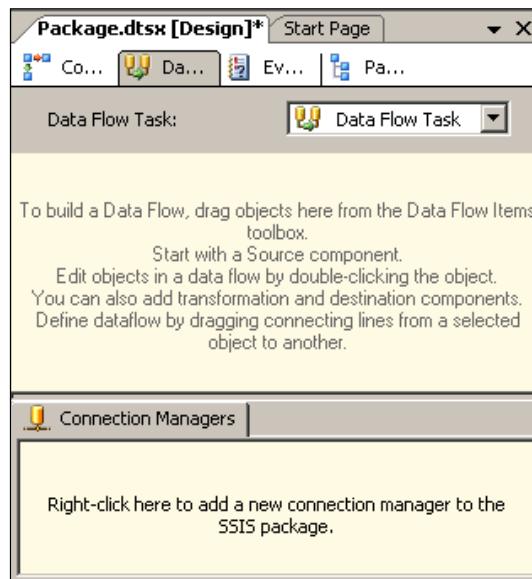
Each of these menu items would open up windows where you can set up the connection used by **Source**, **Destination**, and sometimes **Data Flow** transformation. If the type of connection you are planning is not on the menu list, you can click on **New Connection...** and configure your desired connection. In this chapter, we will not be using the Connection Manager.

Data Flow

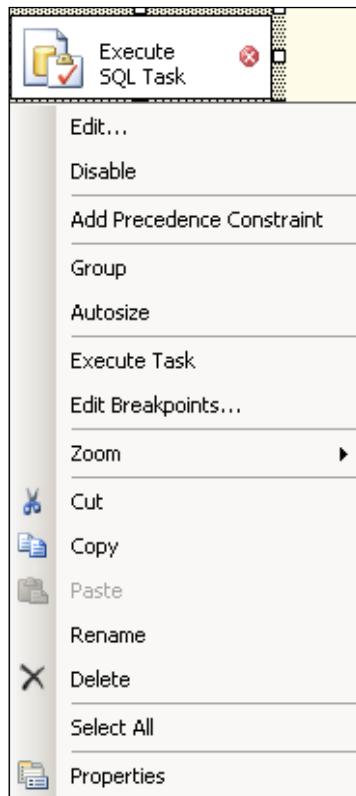
Clicking on the **Data Flow** tab will take you to the **Data Flow** page, as shown in the following screenshot.



You may click on "**No Data Flow tasks have been added to this package. Click here to add a new Data Flow task.**" to add a Data Flow task. The page changes to this:



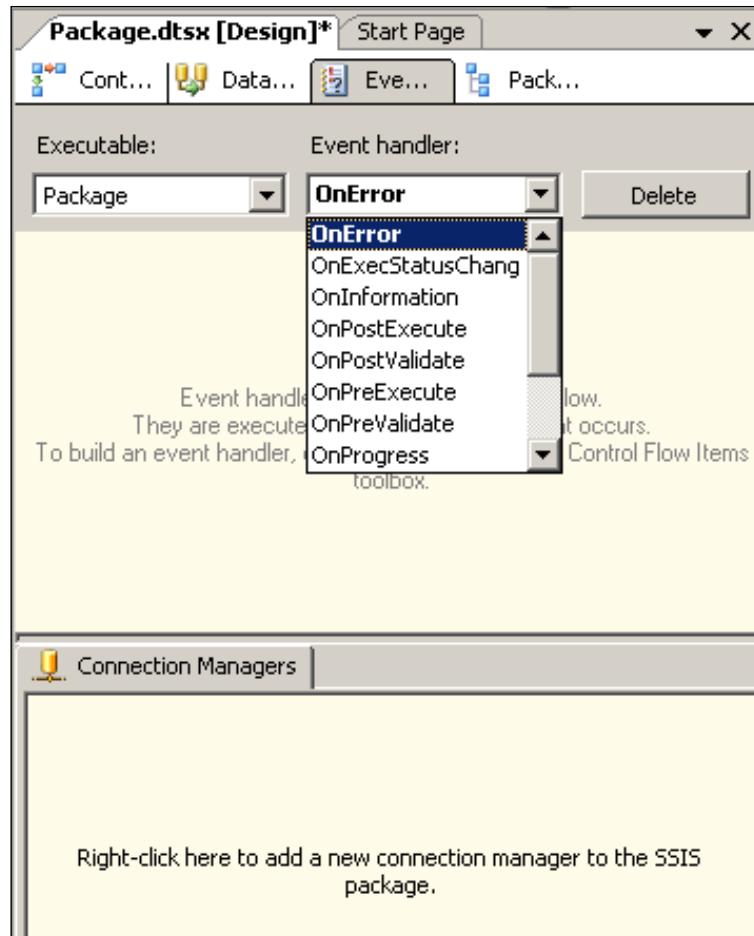
Read the instructions for building a Data Flow, given on this page. Data needs to flow from a source to a destination after going through the transformations stage. By double-clicking on **Data Flow** items in the toolbox, you can add items to this page. Once the objects are added, you can edit their properties by right-clicking on the object, and then clicking on **Edit**. The **Connection Manager** page is available here as well. After this page is displayed, if you click on the **Control Flow** tab, you will see a **Data Flow Task** added there as well. In fact at this point, you should be seeing an **Execute SQL Task** and a **Data Flow Task** on your **Control Flow** page. It is as easy to delete objects, as adding objects by right-clicking the object and choosing to delete it as shown in the next screenshot where **Execute SQL Task** can be deleted.



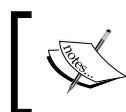
There are several other menu items such as **Execute Task**, **Edit Break Points...**, **Add Precedence Constraint**, etc. All these are useful in the package design.

Event Handlers

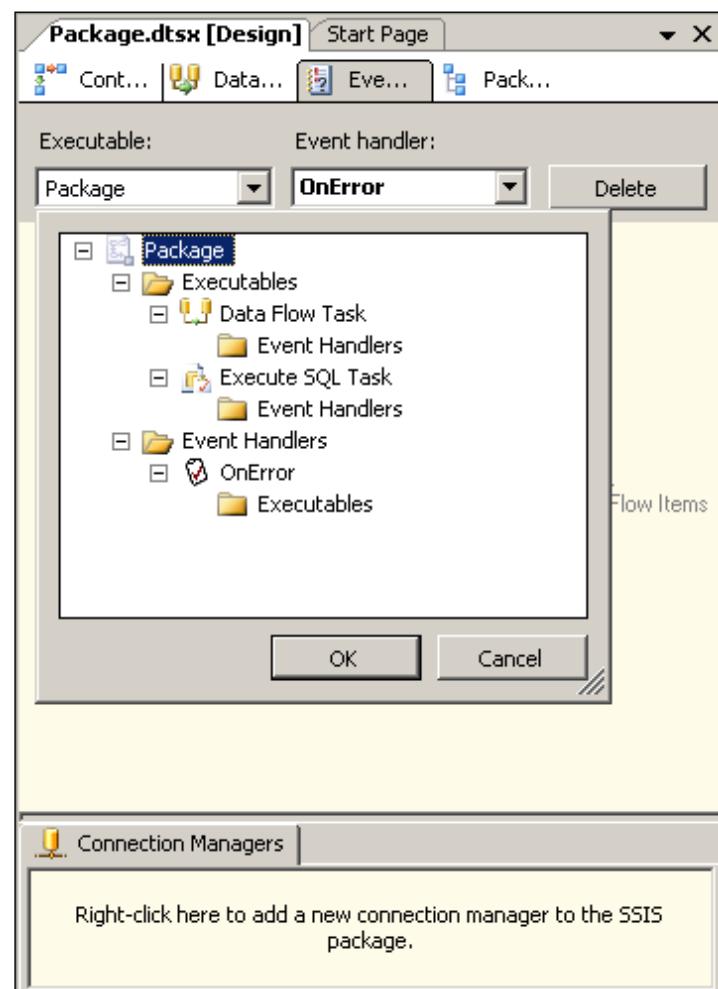
The Event Handler page can be displayed by clicking on the **Event Handler** tab. This brings up the **Event Handler** page as shown in the next screenshot. The top object for this page is the **Package** itself as shown with the default event, **OnError**. Of course there are other event handlers to choose from the drop-down list.



A package is not the only object that can raise events. Objects contained in the package can also raise events. This can be seen by clicking on the drop-down for **Executable** in the above screenshot. In the next screenshot, **Execute SQL Task**, and **Data Flow Task** are both executables and have **Event handlers** associated with them, with which you can trap events.

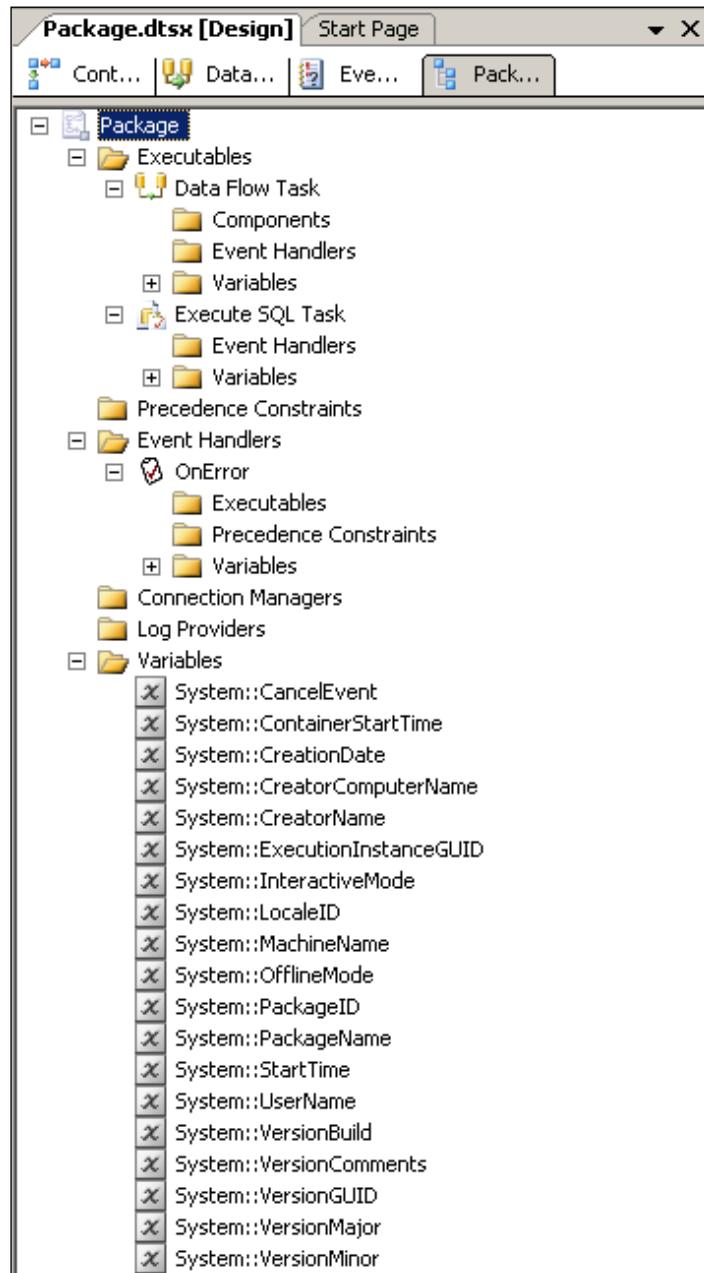


Do not skip reading instructions on several of the windows that you will be using as they give you very specific instructions as to what you may do on that page.



Package Explorer

Click on the **Package Explorer** tab to reveal everything that pertains to the package, as shown below.



This shows the complete details of the package. It shows two tasks, **Execute SQL Task** and **Data Flow Task**, and as these may execute in a given order there is a folder for **Precedence Constraint** which executes first. **Data Flow Task** is going to have other components, and therefore there is a folder for components. Each of these executables can have their own variables including the package. There are event handlers with each of these executables.

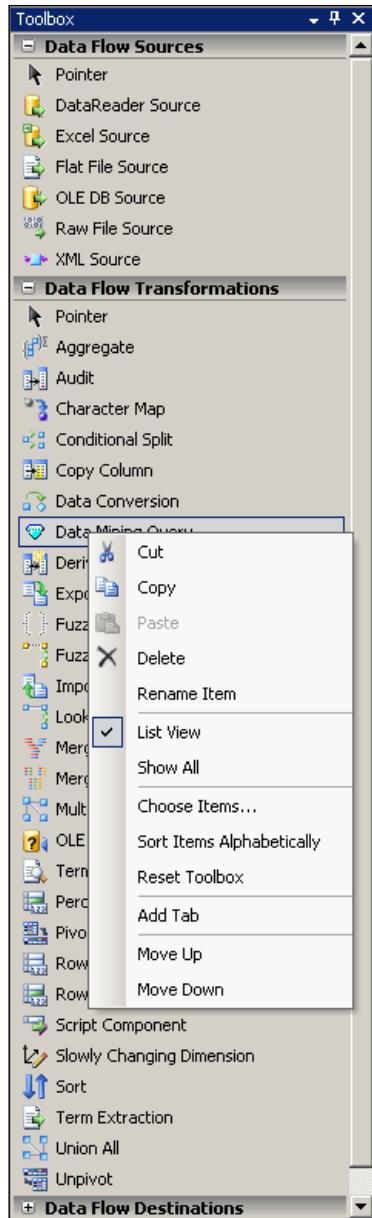
The Toolbox

If you are on the **Control Flow** page of the canvas, you will see **Control Flow Items** and **Maintenance Plan Tasks**. On the other hand, if you are on **Data Flow** page, you will see items related to **Data Flow Sources**, **Data Flow Transformations**, and **Data Flow Destinations**. When you are on either **Event Handlers** or **Package Explorer**, you will see **Control Flow** items and **Maintenance Plan** items.

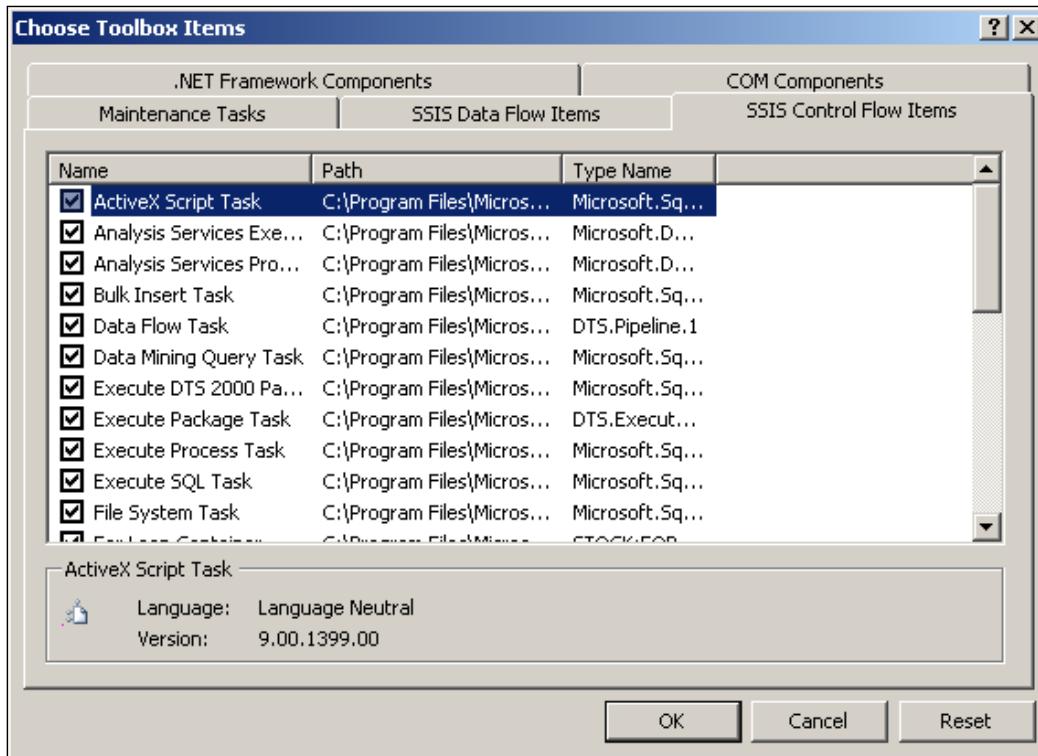
Adding Items not Available in the Toolbox

In a default installation, all the components of the toolbox will be present. These were described in Chapter 1. Should you find any missing for any reason you can add it to the toolbox, as explained in this section.

The toolbox has a drop-down menu associated with it, shown in the next screenshot. This will be displayed when you right-click anywhere in the toolbox.

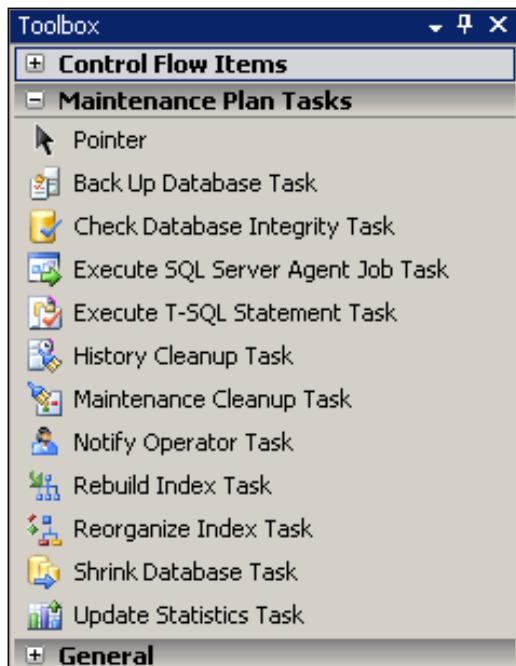


You can customize the toolbox using the drop-down menu items such as **Move Up**, **Move Down**, **Sort**, etc. If an item is not available, you can click on **Choose Items...**, which will bring up a whole lot of objects that you can add as shown. All you need to do is place a check mark and click on **Add**. You may need to scroll up and down to locate the required object. If you are planning to add other items to your solution, you may choose to add **.NET Framework Components** as well. The IDE is well suited to cater to a variety of projects.



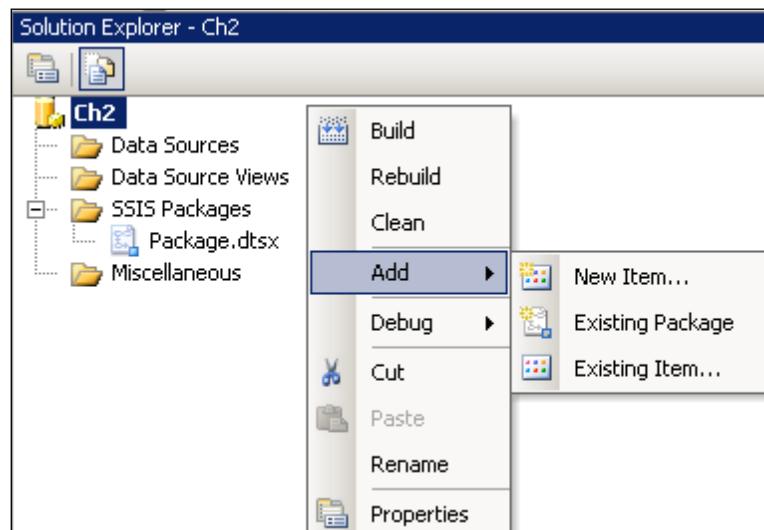
Toolbox Items

The toolbox consists of items related to control flow, data flow, etc. **Control Flow Items** consists of containers and tasks, **Data Flow Components** consists of sources, transformations, and destinations. This window is available in the context of the **Control Flow** page. The tasks shown here are the usual tasks undertaken by a 'DBA'. As tasks are related to a database, you generally need a connection manager to point to the database on a server.

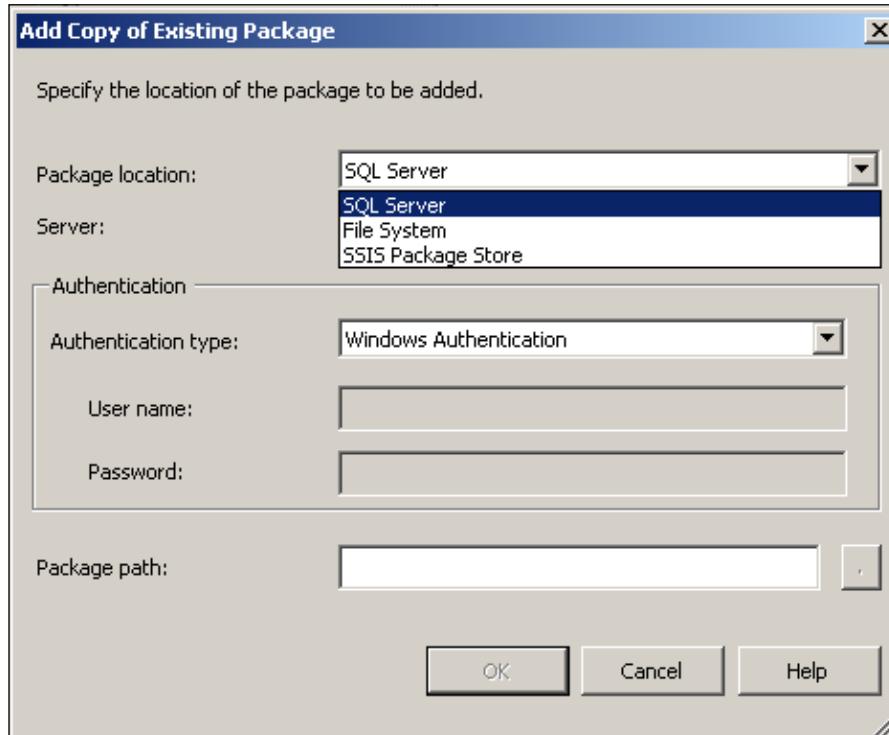


The Solution Explorer

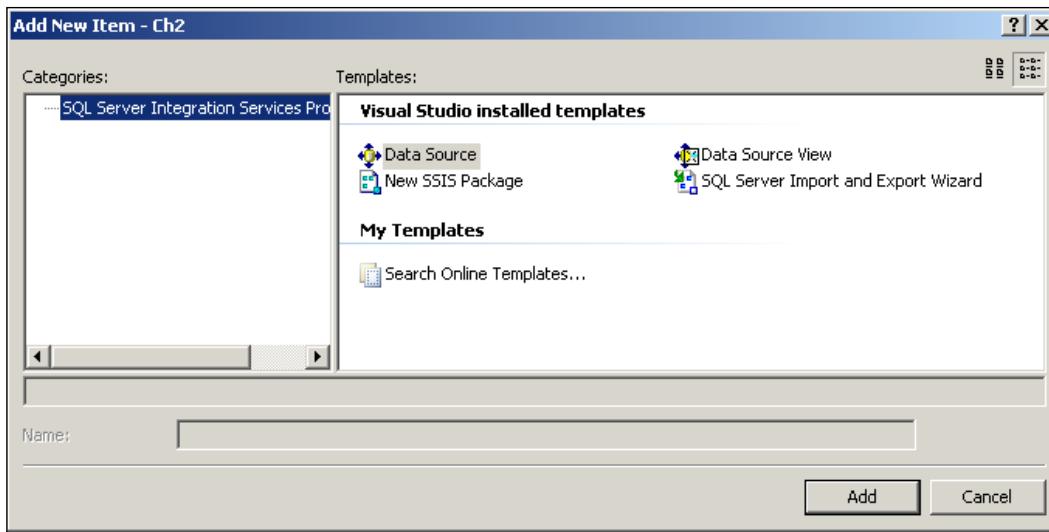
As the name suggests, the **Solution Explorer** window shows every project that is a part of the solution. You can add other projects, like an Analysis Services project, a Report Services project, etc. The **Solution Explorer** is a feature-rich interface with many menus tucked inside it. When the project is created for the first time, this window is populated by a tree view showing the various items created, as shown below. You can use this drop-down menu to bring in an existing package, a **New Item** or an **Existing Item**. There is a lot of flexibility built into this IDE. You can also **Build**, **Rebuild**, or **Clean** the project.



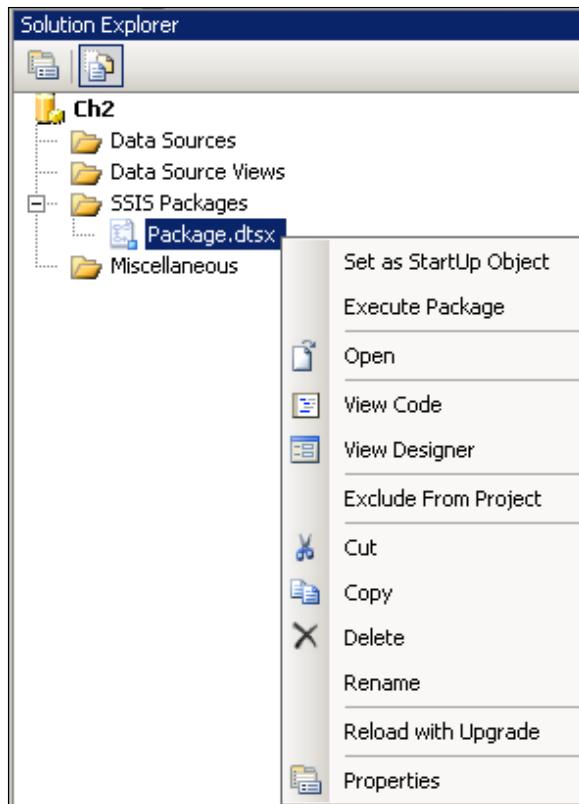
If you choose the **Existing Package** option, this is what will be displayed. Packages can be saved to any one of the three places, although the VS 2005 IDE saves the package to a folder. The window that opens up is now going to look into those places as shown. Of course packages have ownership and you need to get properly authenticated; and you should have the permissions to work with the package. You can also browse to discover the path where you have saved the package.



On the other hand if you had chosen **New Item....**, you would open up a different window. You can add any of the items shown that include a new SSIS package. **Data Source** when selected will bring up its own wizard which will guide you to add a new data source. Similarly, selecting **Data Source View**, will bring up its wizard which will guide you to create a **Data Source View**, which you can add to your project. The wizard is a macro, built for user a interface that helps in the configuration of these objects.



As mentioned previously, it is important to look out for contextual help. Instead of the project if you were to highlight the package and double-click on it, there will be a drop-down menu as shown in the next screenshot:

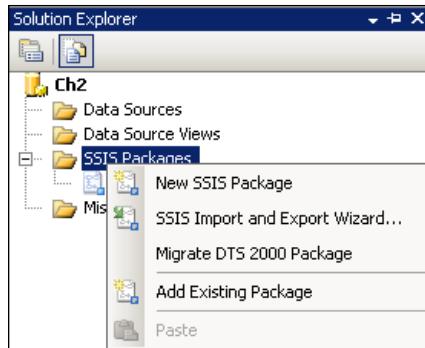


This has a wealth of information. You can execute the package from here by just choosing the **Execute Package** option. If everything is working correctly, you should be able to successfully execute the package. From here, you can see the code or the designer depending on your choice. The package file is an XML document, which you see when you choose **View Code**, consisting of two tasks that are not configured as yet.

```
<?xml version="1.0"?><DTS:Executable  
xmlns:DTS="www.microsoft.com/SqlServer/Dts"  
DTS:ExecutableType="MSDTS.Package.1"><DTS:Property  
DTS:Name="PackageFormatVersion">2</DTS:Property><DTS:Property  
DTS:Name="VersionComments"></DTS:Property><DTS:Property  
DTS:Name="CreatorName">HODENTEK\Jay</DTS:Property><DTS:Property  
DTS:Name="CreatorComputerName">HODENTEK</DTS:Property><DTS:  
Property DTS:Name="CreationDate" DTS:DataType="7">6/5/2007  
11:04:42 AM</DTS:Property><DTS:Property  
DTS:Name="PackageType">5</DTS:Property><DTS:Property  
DTS:Name="ProtectionLevel">1</DTS:Property><DTS:Property DTS:Name=  
"MaxConcurrentExecutables">-1</DTS:Property><DTS:Property  
DTS:Name="PackagePriorityClass">0</DTS:Property><DTS:Property DTS:  
Name="VersionMajor">1</DTS:Property><DTS:Property  
DTS:Name="VersionMinor">0</DTS:Property><DTS:Property  
DTS:Name="VersionBuild">1</DTS:Property><DTS:Property  
DTS:Name="VersionGUID">{B391A52D-01A3-44FA-8E81-AC70B672E6D6}</  
DTS:Property><DTS:Property  
DTS:Name="EnableConfig">0</DTS:Property><DTS:Property  
DTS:Name="CheckpointFileName"></DTS:Property><DTS:Property  
DTS:Name="SaveCheckpoints">0</DTS:Property><DTS:Property  
DTS:Name="CheckpointUsage">0</DTS:Property><DTS:Property  
DTS:Name="SuppressConfigurationWarnings">0</DTS:Property>  
<DTS:PackageVariable><DTS:Property DTS:Name="PackageVariableValue"  
DTS:DataType="8">  
<!--portions removed  
<DTS:Property DTS:Name="Description"></DTS:Property><DTS:Property  
DTS:Name="CreationName">OnError</DTS:Property></DTS:EventHandler></  
DTS:Executable>
```

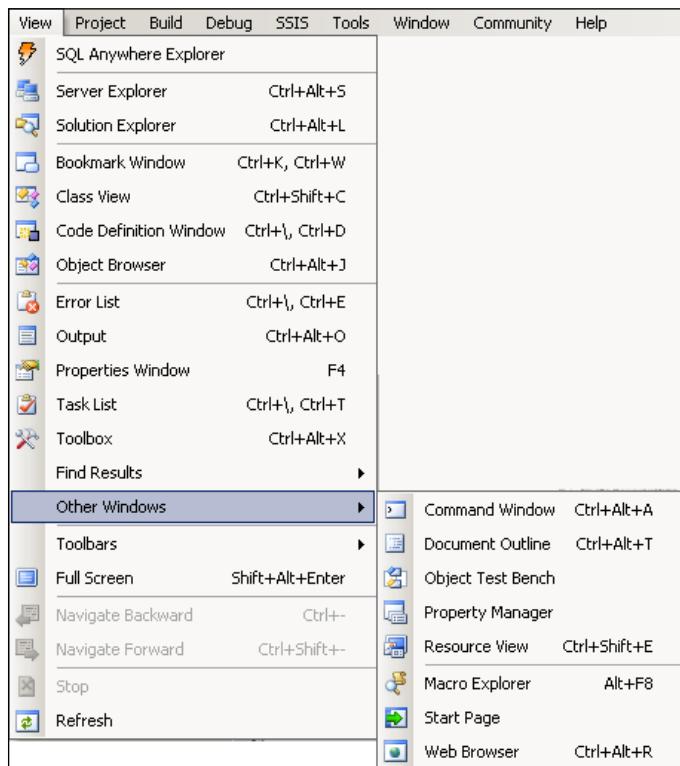
You may even choose to exclude this package from the solution and focus on another.

The **SSIS Packages** folder has its own drop-down as shown in the next screenshot. Here you see even more SSIS functionality. You can bring up an export/import wizard which will guide you through the data transfer process. You may even choose to migrate a stored package designed using the DTS and stored in SQL Server 2000. Of course, leaving aside the default package you can start a new package. **SSIS Import and Export Wizard...** can also be invoked from inside SQL Server 2005 management studio by right-clicking on the management node and choosing **Export** or **Import** menu items (see Hands-On Exercise 3 Invoking Export, Import Wizard in SQL Management Studio).



Getting Various Windows

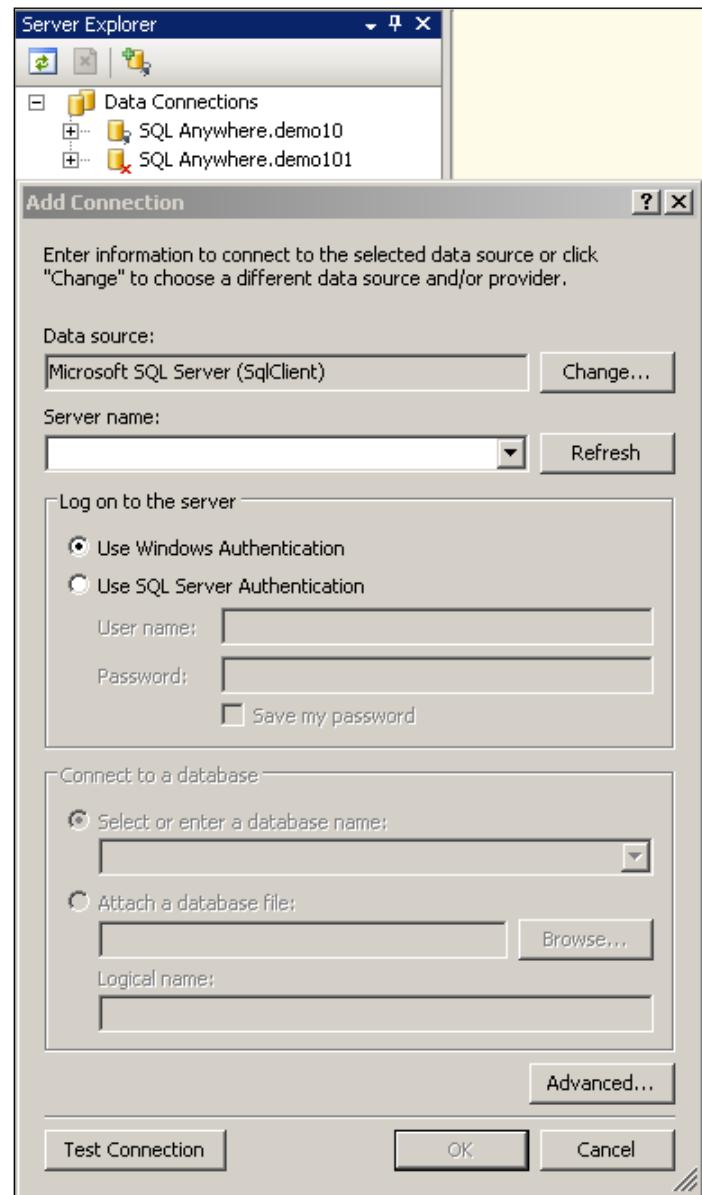
Besides the various windows we have seen so far, there are many windows available for various kinds of tasks. All these can be accessed by clicking on the main menu item, **View**. This brings up a drop-down menu, as shown below:



The various windows provide information regarding objects, processes, errors, properties, etc.

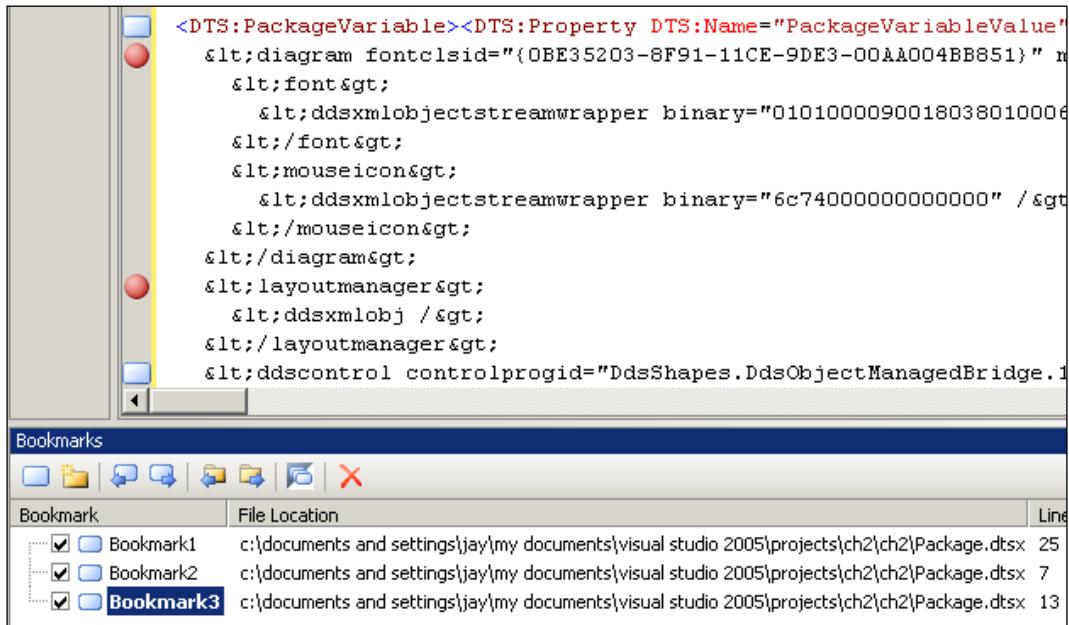
Server Explorer Window

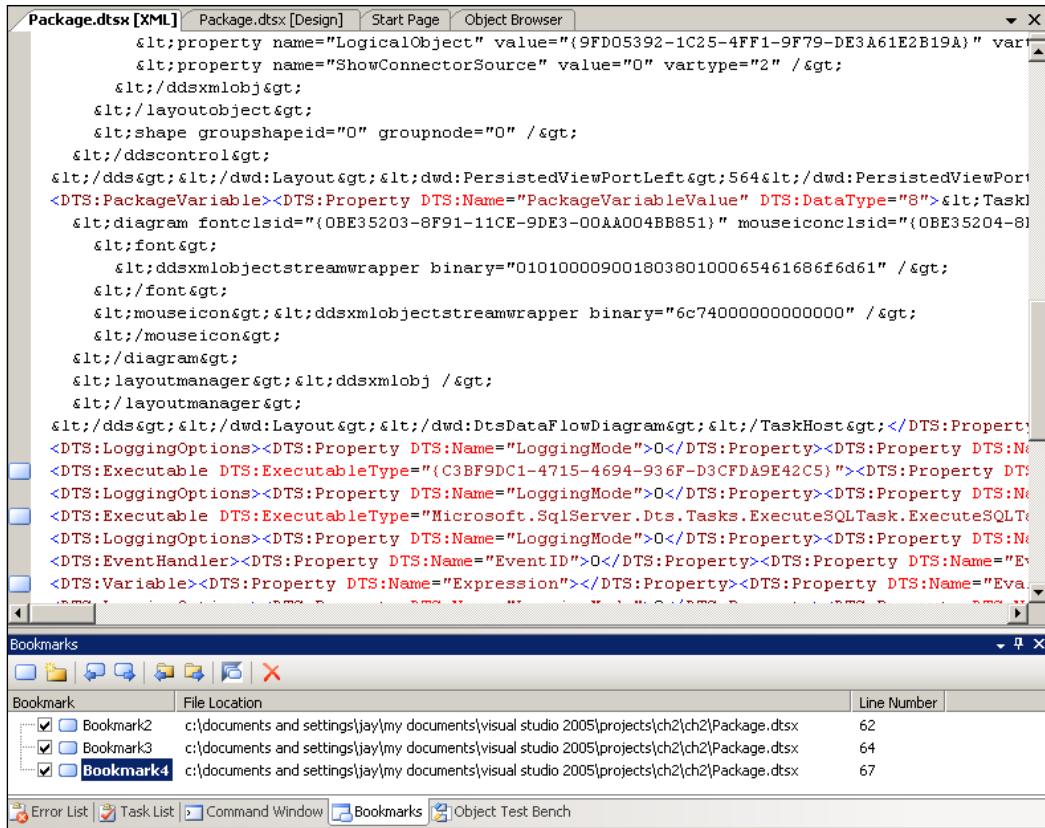
Server Explorer shows you the data sources to which you are connected. You can establish a connection by right clicking on the connection and choosing **Add Connection...** which brings up the window shown next. Right now it is showing one active connection and one inactive connection. You could connect to several sources using this window. We will see how in the following chapters.



Bookmark Window

You could bookmark lines of code and get to see them later. The following screenshot shows two of the three bookmarks on the code and how they get saved in the Bookmark window. Place your cursor on the line of code and click on the rectangular icon in the **Bookmark** window. The line gets a reference and the line in the code gets an icon along side of it.



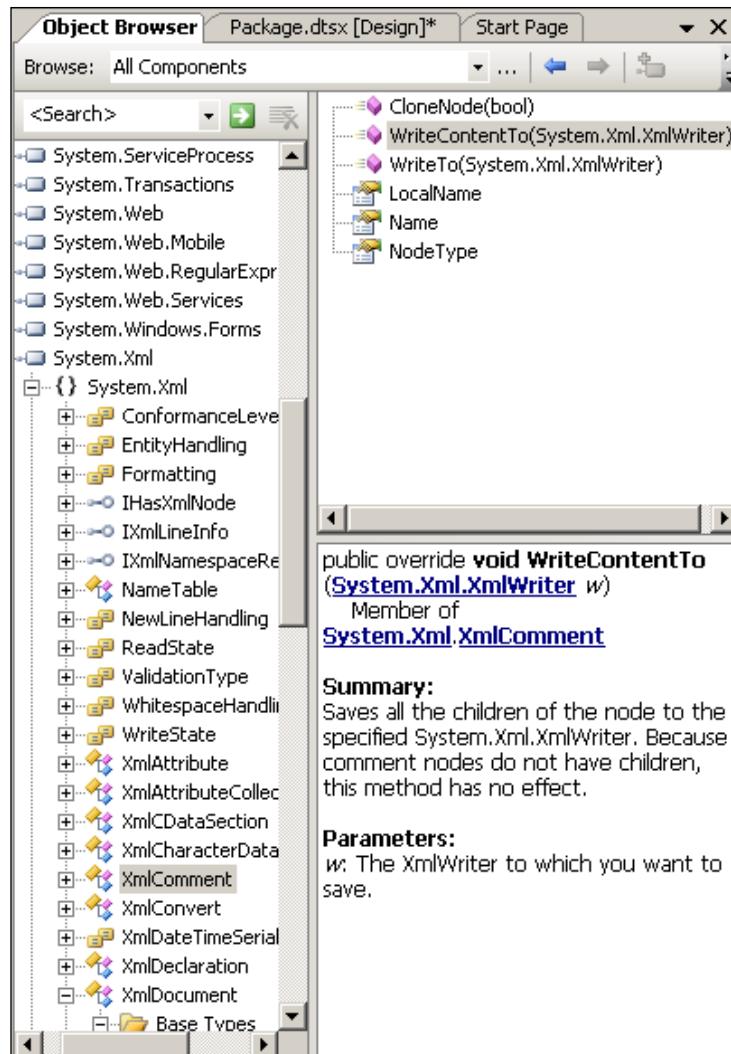


Class View and Code Definition Windows

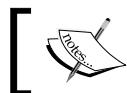
These windows are empty, unless you are dealing with class files and code definitions. This is actually a part of the Visual Basic .Net tool. Since it is an IDE for several project types, these windows will be useful for those applications that use them specifically.

Object Browser

This is a very important window for this IDE. You can obtain a fine grained definition or source of any of the .NET framework classes here, as shown in the next screenshot:



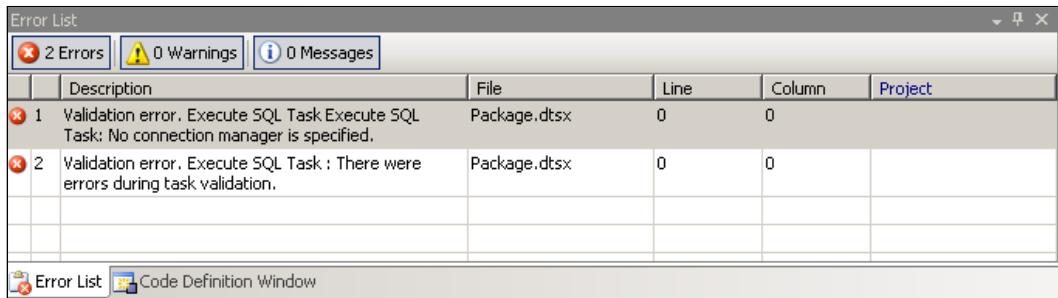
In the previous screenshot, you saw the **WriteContentTo** method going back to the **System.XML** namespace, to which the **XMLComment** class belongs.



Object Browser is a very useful window when you are looking for some information on classes and namespaces.

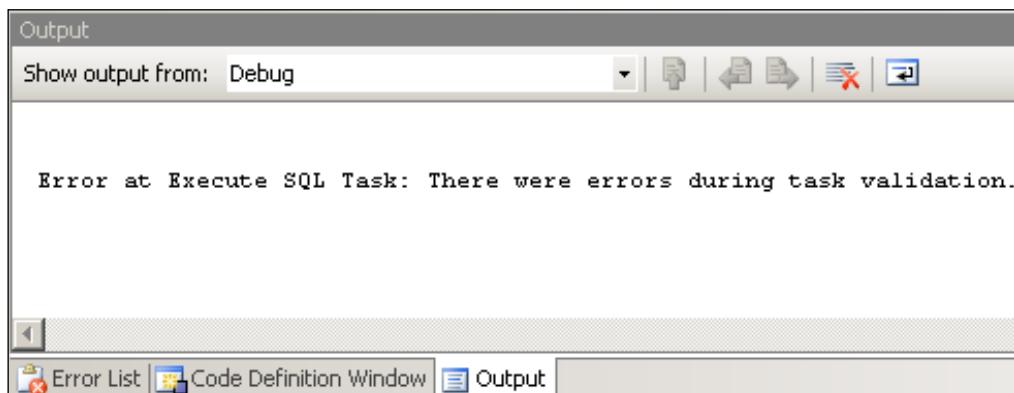
Error List Window

There may be errors in the package, such as a package which is not configured properly, or a variable is not defined correctly, etc. These show up in the **Error List** window, as shown below. You may also get warnings (not as serious as errors) and some messages in this window.



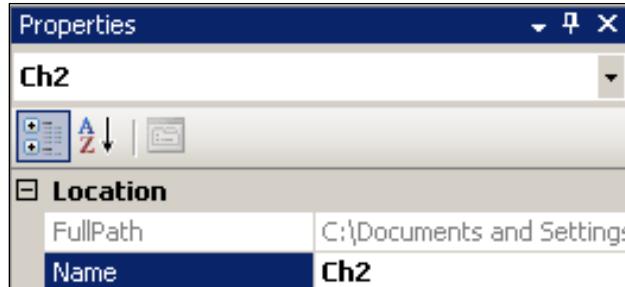
Output Window

This shows all processes that begin when you execute a package. If a package is successfully executed, it shows all phases in its execution. When you try to execute a package, you get a message in the output window. The following error is produced when you try to execute the package in the project **Ch 2**.

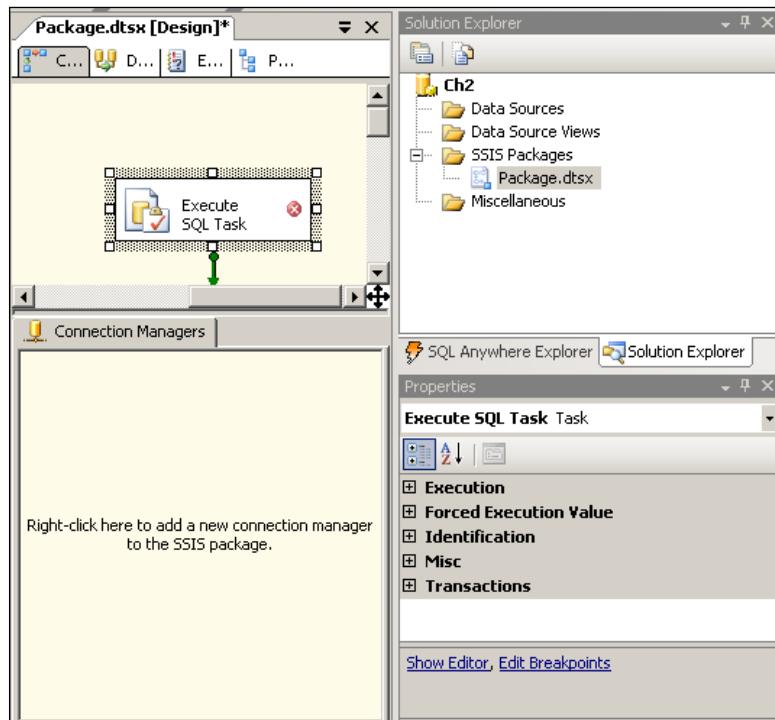


Properties Window

The **Properties** window of the project shows, where the project is located on the machine, which is given by the **FullPath** field in the **Properties** window, as shown. In addition to the **Properties** window for the project, there is also a **Property Pages** window associated with the project.



If you first click on the **Control Flow** page in the designer, and click on the **Properties** sub-menu under **View**, you will see the properties of the package as shown. The context is important here. This window can be displayed by other means as well. For example, when the **Control Flow** window size is such that it does not show all the objects contained therein (there are two tasks of which one is hidden due to the reduced size of the Control Flow Page), a cross-mark will be displayed (see the following screenshot). Clicking this button with the symbol opens up the properties of the package, assuming you have chosen the **Properties** sub-menu in the **View** menu. The drop-down menu of the **Package** shows all the items in the package container. From here you can display the hidden items, by clicking on an item in the **Package** drop-down menu.



Tasks List Window

This is a window where you can create and keep track of tasks. It is quite similar to a "to-do" list. You can add a task by clicking on the icon with a right check mark in red next to the drop-down menu. You can add a task to this list, as shown here:



Toolbox Window

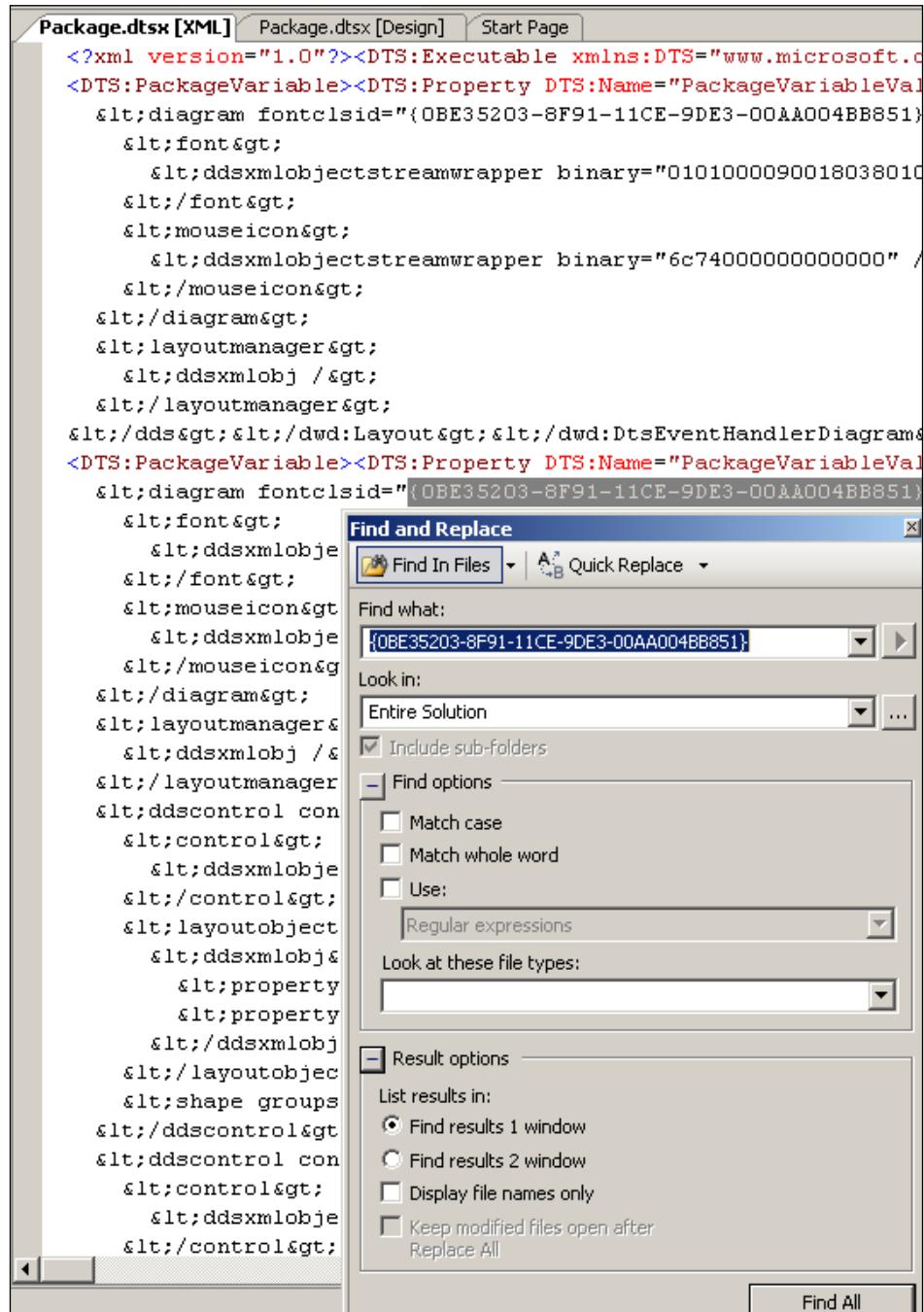
This window is customizable, and depends on the **Package Designer** page; when selected, you can see **Control Flow Items & Maintenance Plan Tasks**, or **Data Flow Items**. You can use one or more tool items to design a package.

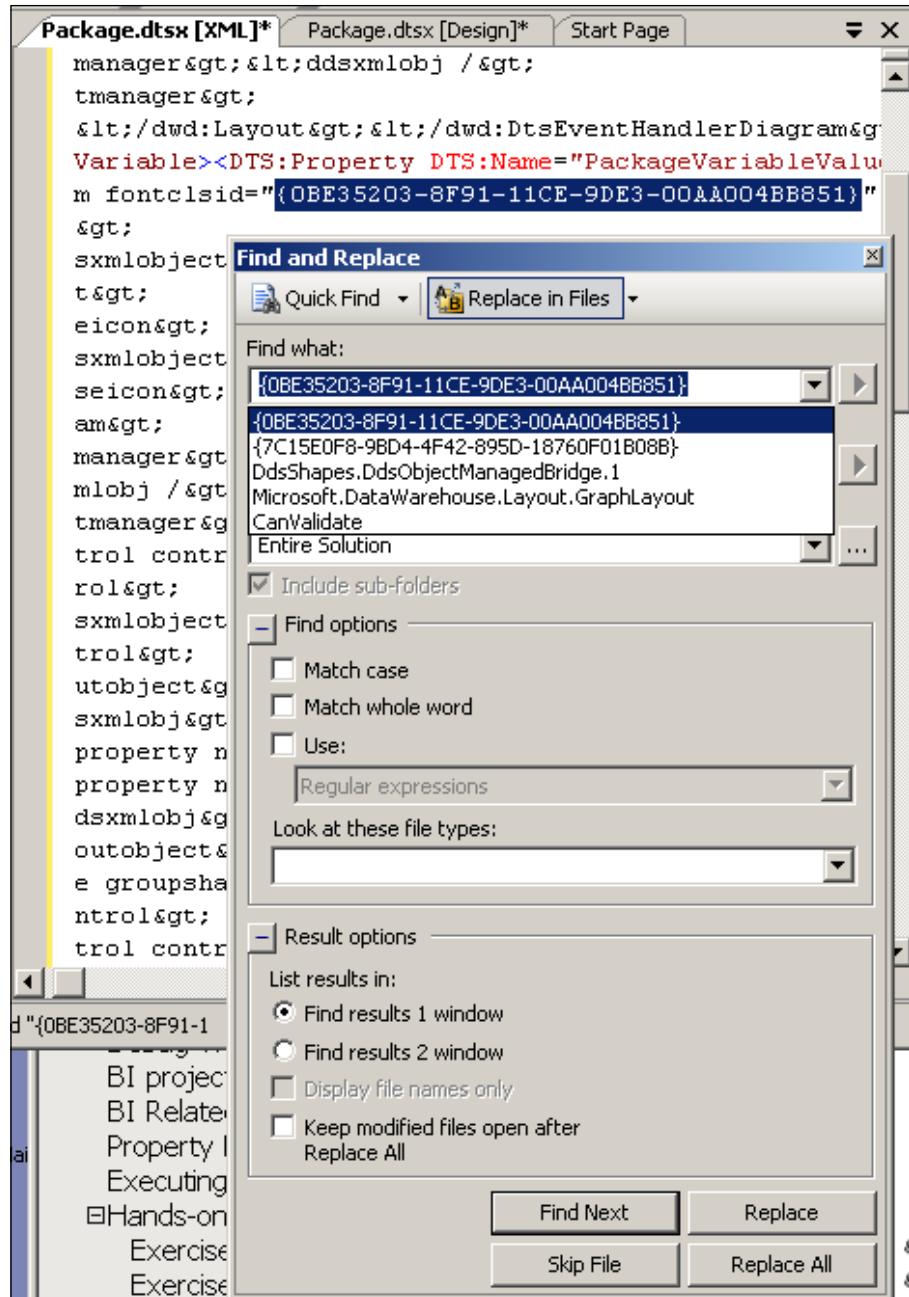
Find Results Window

This window is used when you want to **Find**, or **Find and Replace** items that are normally used while editing in Visual Studio Projects. It is often used along with **Object Browser**. It consists of three sub-menus: **Find Result 1**, **Find Result 2** and **Find Symbol Results**.

Let us say you want to find out about a certain object and its reference. To do this, highlight the object in the code page **Package.dtsx[XML]**, and then from the **Edit** menu, click on **Find and Replace**, and then navigate to the **Find In Files** drop-down menu item.

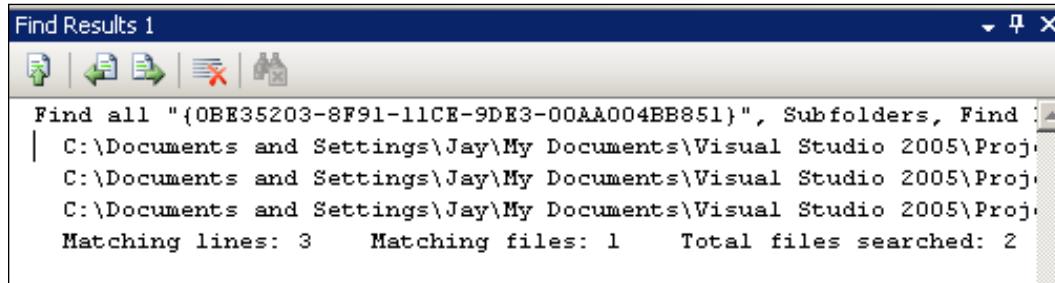
This pops-up a **Find and Replace** window, as shown below.





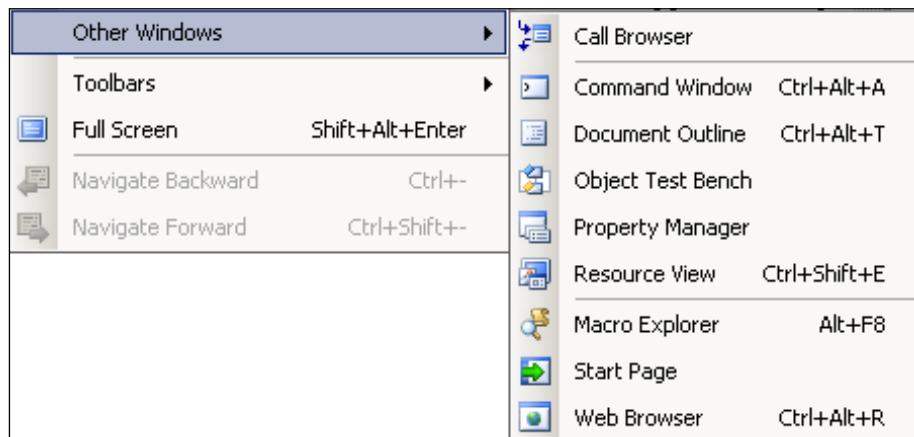
The previous screenshot shows a window with **Find options**, and **Result options** expanded. By default, the results will be shown in the **Find Results 1** window, which opens automatically when the **Find All** button is clicked.

The next screenshot shows the **Find Results 1** window, with the matches displayed in it.



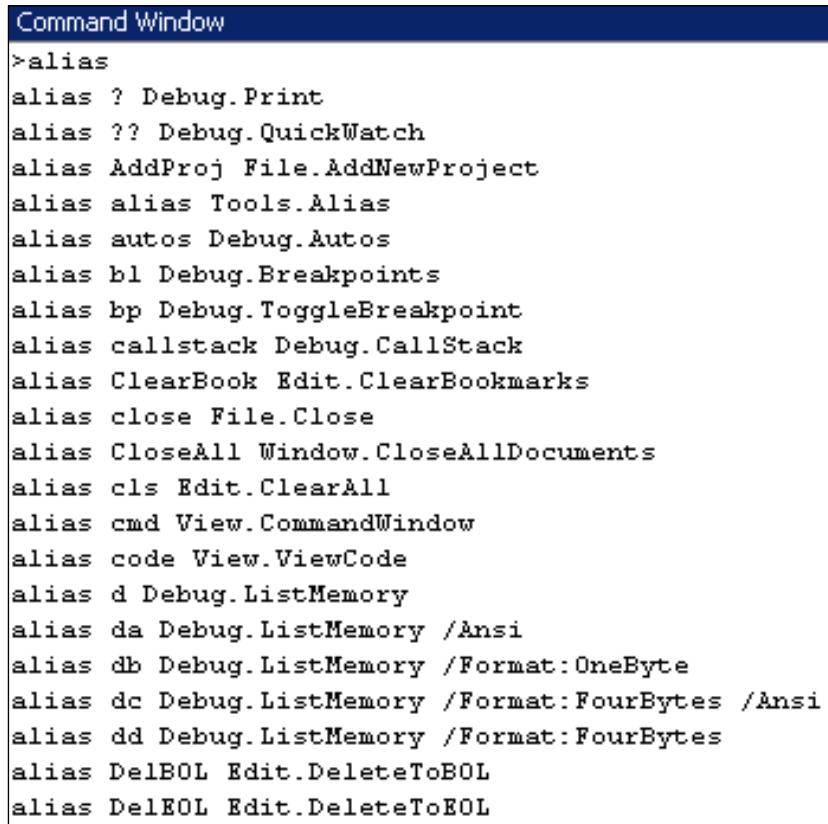
Other Windows

The **Other Windows** sub-menu item opens its own menu. This collects miscellaneous windows.



The **Call Browser** window is mostly used in designing with C++, for locating function calls and navigating to their code in the source.

The **Command Window** is a very useful window as you can carry out a lot of tasks. To get an idea of the commands that can be used, type alias in the output window. This displays items as shown in the following screenshot. If you do find something already in the window, you can type cls and return. This would clear the window. You can use this to go to a certain line in your code, and print the result of a certain variable, etc. If the variable is defined for one of the objects, the debug.print <space>variable will execute the package to get to the result. By typing in code after the > symbol, you can open up the code page since **code** is an alias for **View. Viewcode**, as shown below. You can also start executable programs like notepad, by typing Tools.shell notepad after the > symbol. Try to spend some time in this window to get to know the details. While your cursor is in the **Command Window**, you can press *F1* for help.



The screenshot shows a Windows command-line interface window titled "Command Window". The window contains a list of command aliases, each starting with ">alias" followed by the alias name and its corresponding command. The list includes:

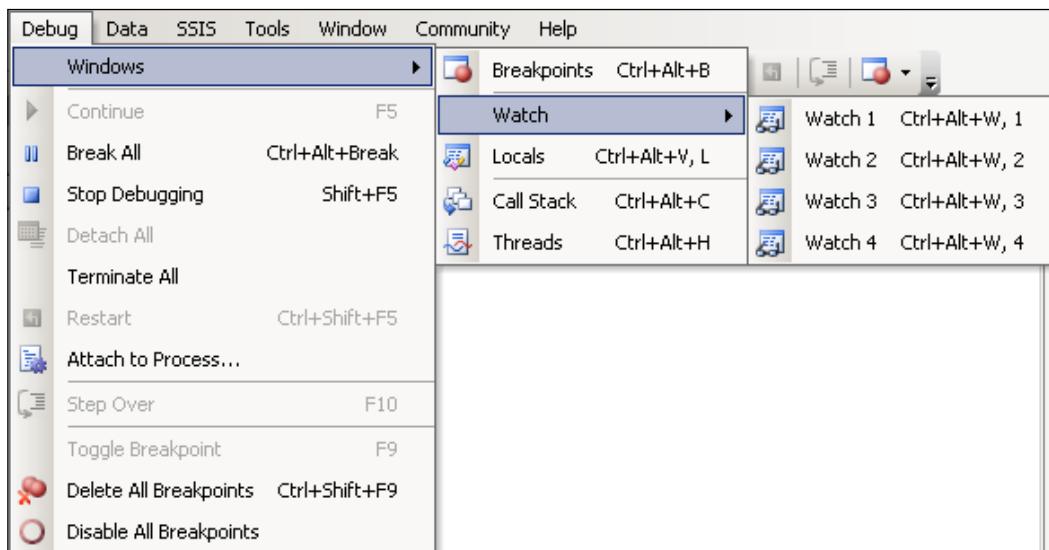
- >alias
- alias ? Debug.Print
- alias ?? Debug.QuickWatch
- alias AddProj File.AddNewProject
- alias alias Tools.Alias
- alias autos Debug.Autos
- alias bl Debug.Breakpoints
- alias bp Debug.ToggleBreakpoint
- alias callstack Debug.CallStack
- alias ClearBook Edit.ClearBookmarks
- alias close File.Close
- alias CloseAll Window.CloseAllDocuments
- alias cls Edit.ClearAll
- alias cmd View.CommandWindow
- alias code View.ViewCode
- alias d Debug.ListMemory
- alias da Debug.ListMemory /Ansi
- alias db Debug.ListMemory /Format:OneByte
- alias dc Debug.ListMemory /Format:FourBytes /Ansi
- alias dd Debug.ListMemory /Format:FourBytes
- alias DelBOL Edit.DeleteToBOL
- alias DelEOL Edit.DeleteToEOL

Document Outline, Object Test bench, Property Manager, Resource View, and Macro Explorer windows are windows mostly used in various other VS projects. They are all accessible if your solution adds other project types to the solution.

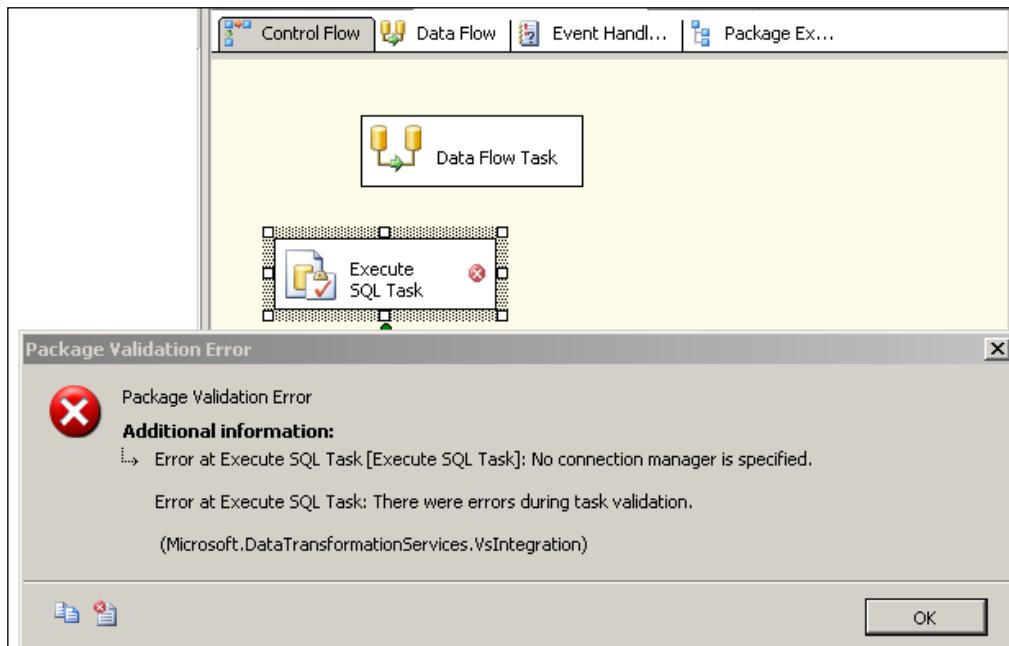
The **Start Page** sub-menu item takes you to the **Start Page** which appeared when the application was launched, while the **Web browser** sub-menu item takes you to the Microsoft Web site <http://www.microsoft.com/sql/default.mspx> for SQL Server.

Debug Windows

Debug windows are not accessible from the **View** menu. When the package is executed, and the program is in the debug mode, you can see all sub-menu items of the **Debug** menu. While the program is running in debug mode, you will not be allowed to remove objects or close the design windows. There are several important windows used in debugging – **Breakpoints, Watch, Locals, and Call Stack**.

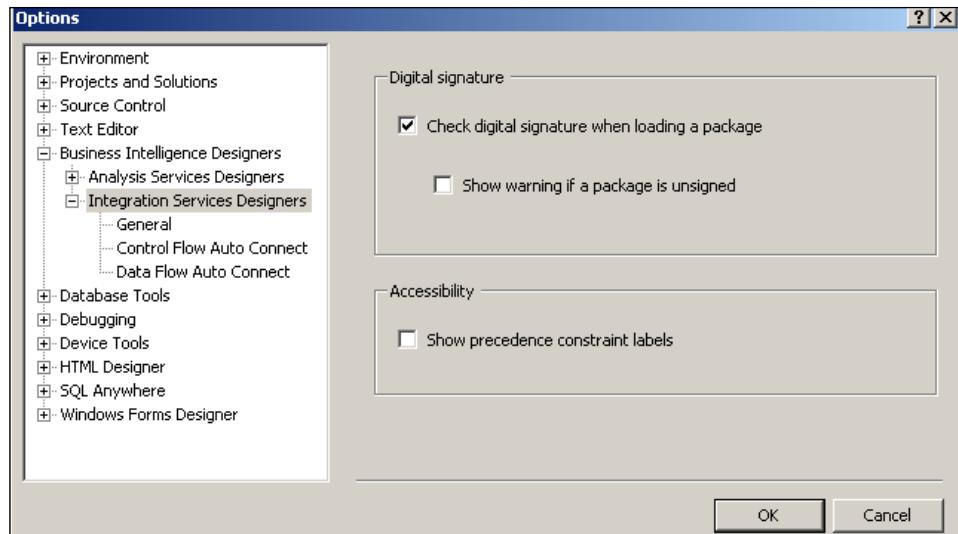


In the Ch 2 project, **Execute SQL Task** does not have a connection manager, and if you try to execute the package, you immediately get an error like that shown below. In this scenario you will not be able to access any of the debugging windows. You can test this by deleting **Execute SQL Task**. Click on **Build** to build a project and execute it. After doing this, you can access all the sub-menus of the debugging window from the **Debug** menu.



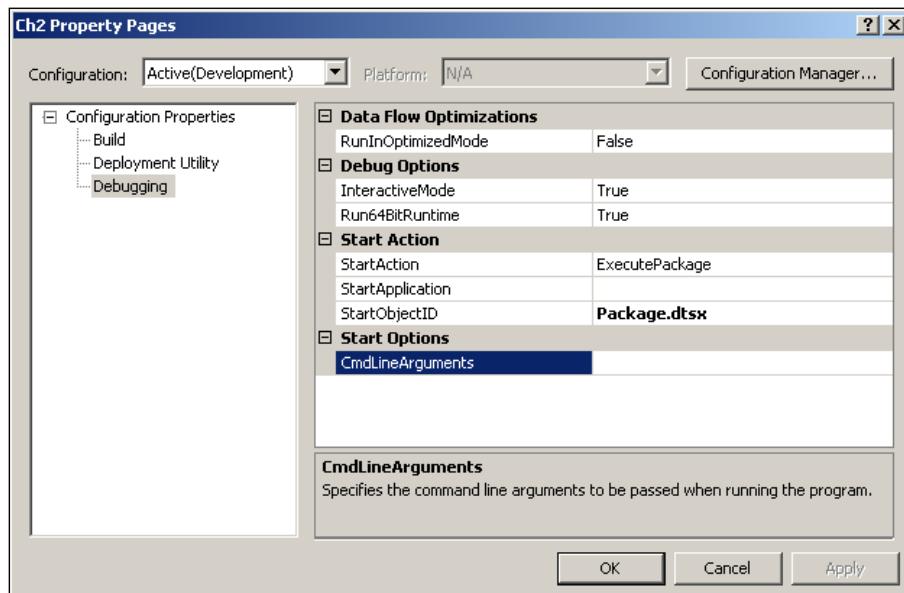
BI Related Items in Tools/Options

There are a couple of items related to SSIS designer in the **Options** sub-menu of the **Tools** main menu. These options can be used to configure the designer such that when a new item is added to either the **Control Flow** page or the **Data Flow** page, the existing items get connected to the added items automatically. This window can be used to configure the **Digital Signature** and **Accessibility** options, while the precedence constraint labels are displayed as shown in the **General** window. For example, if you are unable to distinguish between red and green (colour blind) and the precedence constraint is green, by checking **Show precedence constraint labels** in the following screenshot, a textual representation is displayed alongside the green line, representing the constraint on the **Control Flow** page.

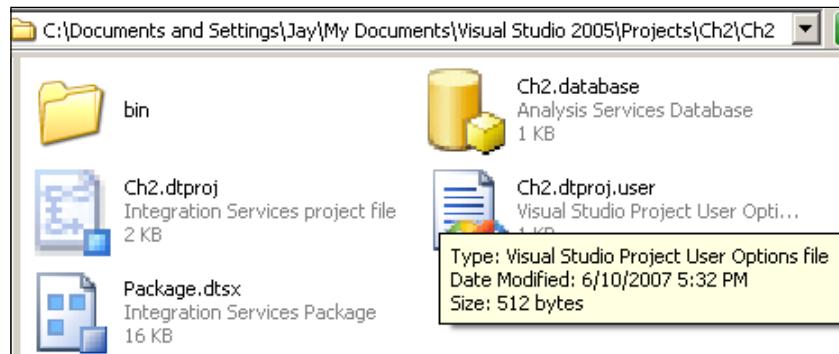


Property Pages and Folders of the Project

While the properties of a project include the name and full path for the project files, the **Property Pages** window shows the various features associated with the project. These pages may be displayed by highlighting the project and right-clicking on it. This opens up the **Property Pages** window (see over page). We will be working mostly in the **Active(Development)** mode with default debugging options on this page.



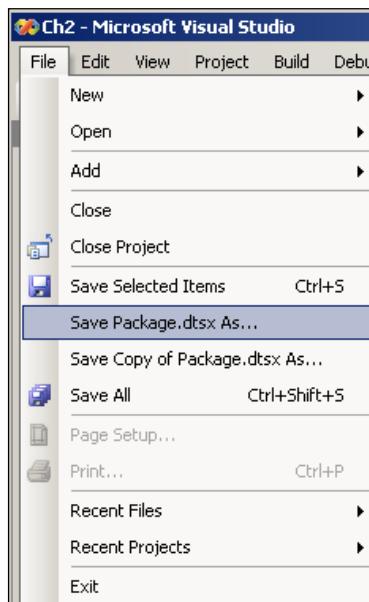
The **Build** option in the above window points the output path to the **bin** directory. The directory structure of this project is shown below:



Executing the Package and Saving the Project

The whole package may be executed by right-clicking on the package (default package in this case) and choosing the **Execute Package** option. If there are multiple tasks on the **Control Flow** page, you may highlight a task and execute it by right-clicking on it and choosing the **Execute Task** option.

The package can be saved to a file system. Hands-On Exercise 2 shows how you may import the package created by the **Ch 2** project to the SQL Server 2005 Management Studio.



Hands-On Exercises

Exercise 1 has been completely described in the main body of this chapter. Please follow Exercise 2 and Exercise 3 step-by-step as described.

Hands-On Exercise 1

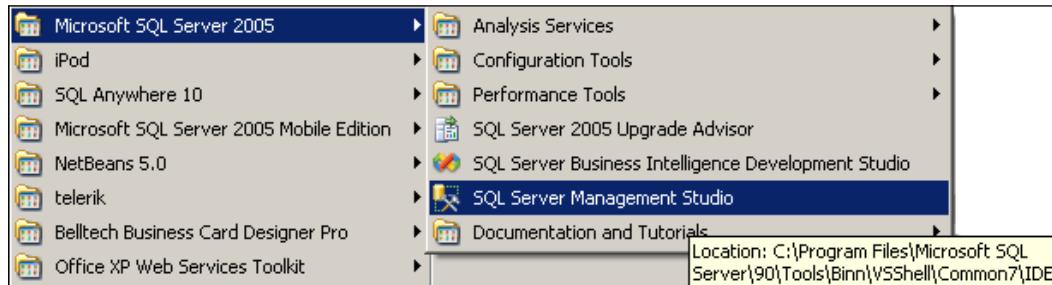
Create a SSIS Project as described in this chapter and get acquainted with the various windows. In doing so, try the following variations:

1. Use the default package and get acquainted with the windows and menus. Rename the package by right-clicking and choosing the **Rename** option. Execute the package (before or after renaming) by right-clicking and choosing the **Execute Package** option. Make sure you notice a tabbed page, **Progress**, added to the Canvas. Now look at the **Progress** page as well as the **Output** window.
2. Try adding different tasks from the **Control Flow** items of the toolbox (both by dragging and double-clicking) and try to understand the added task by right-clicking and looking at the properties.
3. Save the package and explore the folder where it gets saved. Verify if it is at a different location from the one described in this chapter.

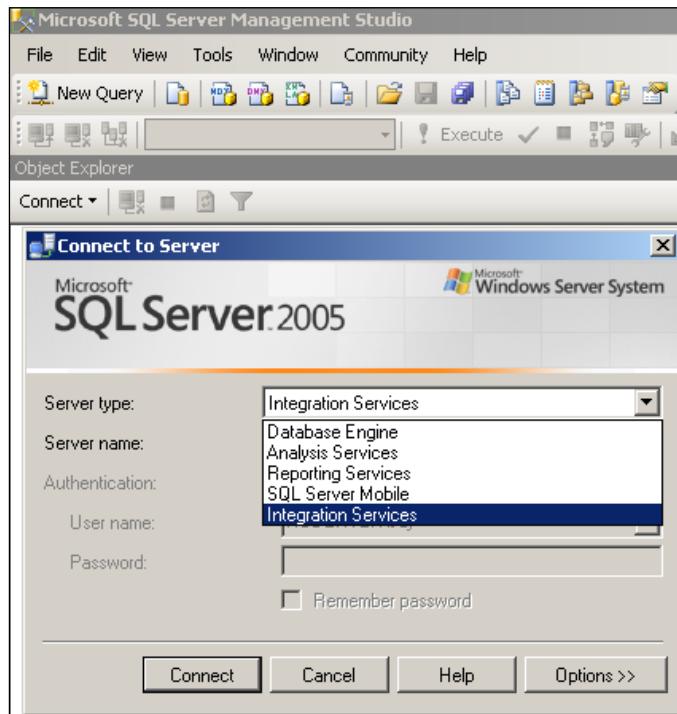
Hands-On Exercise 2

In this exercise, you will import the package that you created in Ch 2 in MS SQL Server 2005 Management Studio.

1. Click and open the **SQL Server Management Studio** by clicking on the shortcut in **All Programs** as shown.

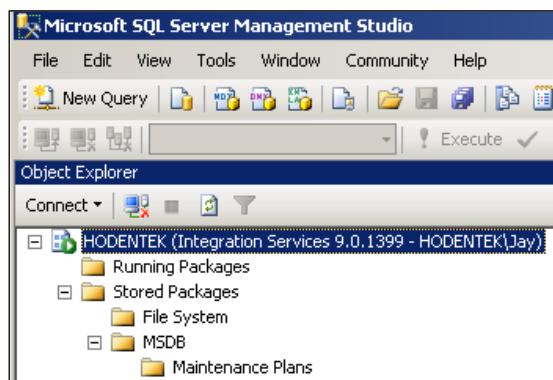


This opens the **SQL Server 2005 Management Studio** below.

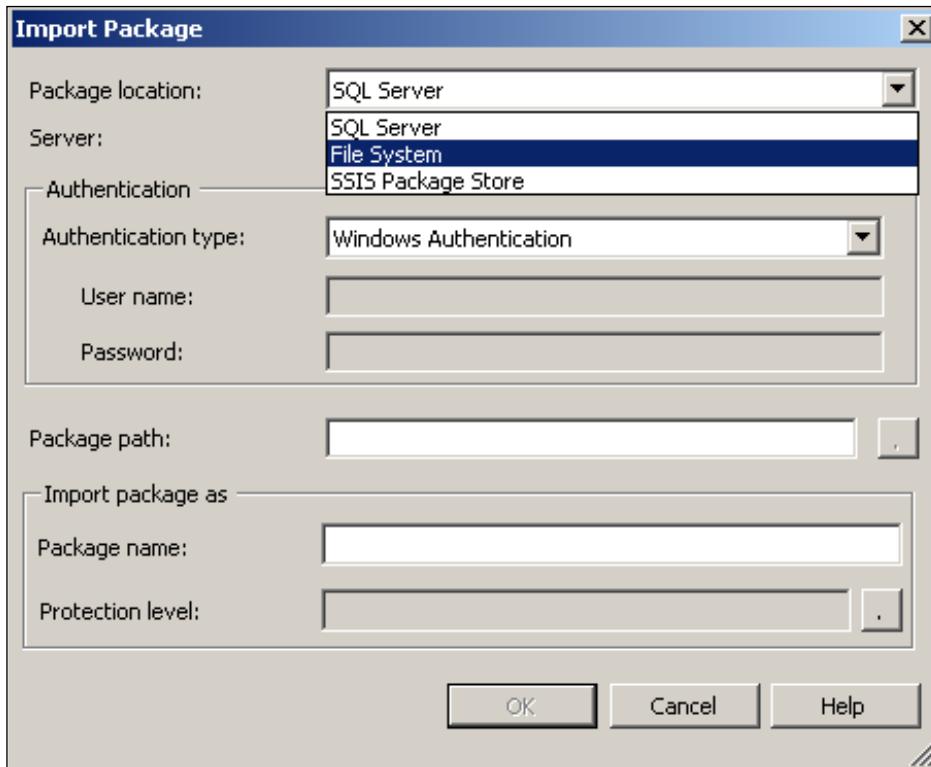


Choose **Integration Services** from the drop-down. It will be showing the configured authentication information. In this case it is authenticated by Windows.

2. Now click on the **Connect** button. This will open the **Integration Service Server** below (showing all nodes expanded out).

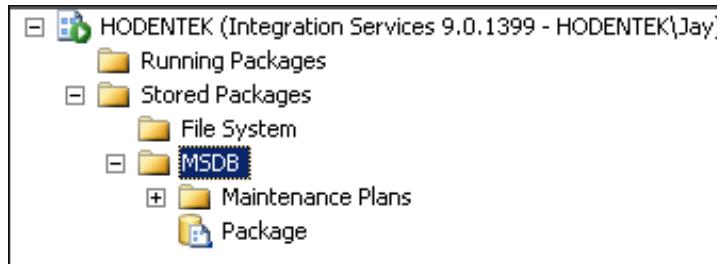


3. Now right-click either on the **File System** folder or on the **MSDB** folder and choose the **Import Package** sub-menu item, which opens the following window. From the drop-down, choose **File System**. For Windows authentication, you do not need **User name** and **Password** when you choose **File System, Package Path**.



4. Now click on the **Package Path** button and browse to the location where it is saved (where the package .dtsx is saved). This adds the package name (in the case of project **Ch 2** the package name is **Package**).
5. Now click on the button **OK**.

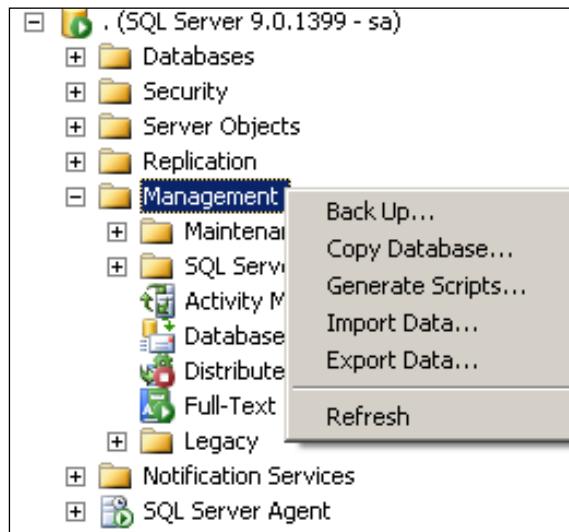
This adds the package to the MSDB folder.



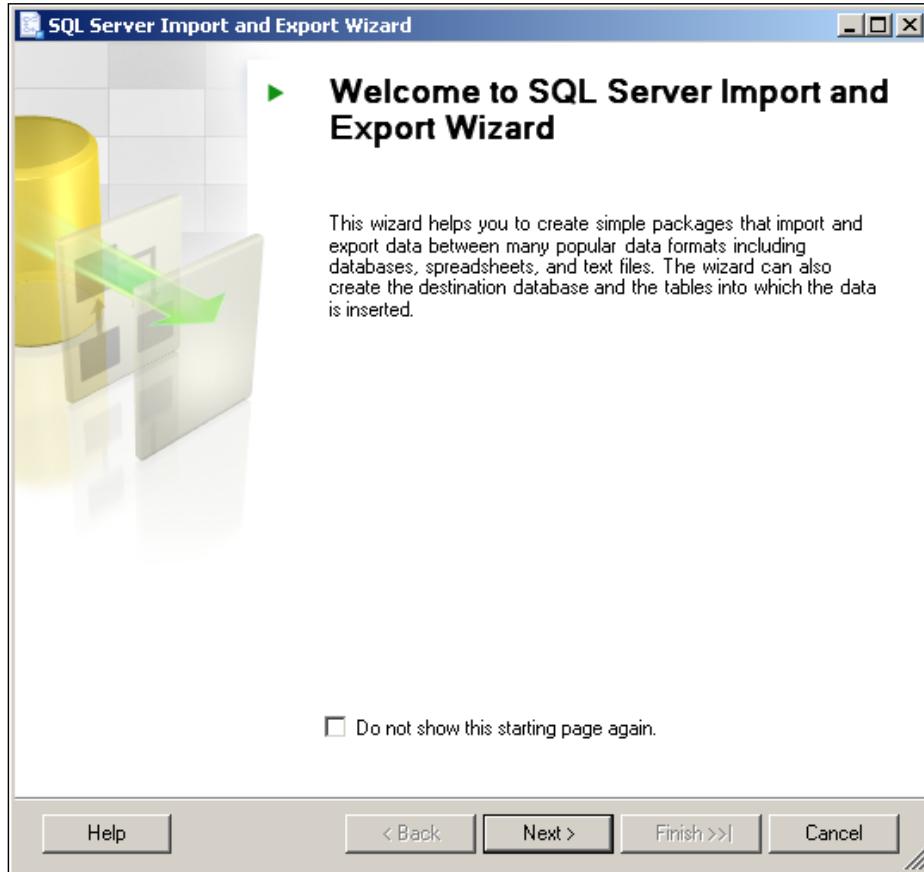
Hands-On Exercise 3

In this exercise, you will be invoking the export/import wizard from the MS SQL Server 2005 Management Studio.

1. Start up the SQL Server 2005 Management Studio.
2. Right-click on **Management**.



3. Now click on either **Import Data...** or **Export Data...** to open the wizard.



Summary

This chapter described the creation of an SSIS project containing a default package, adding and removing objects from the toolbox, and getting acquainted with the various menus, windows, and other project items. Hands-on Exercise 1 helps you get acquainted with the IDE using the instructions in the chapter. Hands-on Exercise 2 shows how you may import the package created by the project into the MS SQL Server 2005 Management Studio. Hands-on Exercise 3 shows how to launch the export/import wizard to create packages in the SQL Server 2005 Management Studio.

3

Sending Email with a SSIS Package

This chapter shows you how to create a SSIS package that can send an email using the Send Mail Task. The Send Mail Task is based on an Internet Protocol called SMTP, which stands for Simple Mail Transfer Protocol.

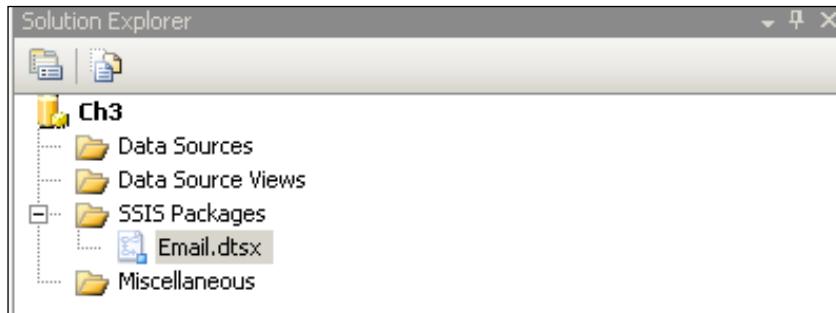
Sending and receiving email is a very common task. The email program on your desktop uses two different servers: one for sending and the other for receiving. The sending of emails is based on the Simple Mail Transfer Protocol described in RFC 821 (<http://www.apps.ietf.org/rfc/rfc821.html>). The Send Mail Task adheres to this standard. Hands-On Exercise One shows how to configure the task and Hands-On Exercise Two shows how to discover your accessible SMTP server. If you are not too sure of an SMTP server that you can use for Hands-On Exercise One, you may read Hands-On Exercise Two first.

Hands-On Exercise One: Sending an Email Using the SMTP Server

For this exercise, you need to know the SMTP server that you can use for configuring this task. You may also use the SMTP server of your web mail program.

1. Create a New BI project **Ch 3** from the **File** menu using the drop-down item, **New**, as described in Chapter 2.
2. In the **Solution Explorer**, right-click **Package.dtsx** and from the drop-down menu choose the option **Rename** and change it to **Email**. Do not change the extension; it should remain **dtsx** (i.e., **Email.dtsx**).
3. You will see a **Microsoft Visual Studio** warning. Click on the **Yes** button on the window with the warning.

The package name changes in the **Solution Explorer**, as shown in the following screenshot, and the package properties are displayed in the properties window.

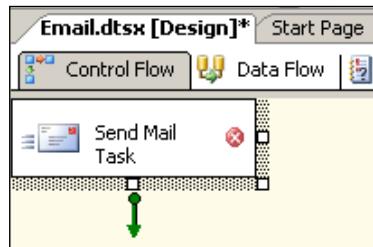


Observe that the Canvas is showing the **Control Flow** tab. This is the **Control Flow** page where you will place your **Control Flow Items**. Read the instructions on this page before you proceed. Now look for **Send Mail Task** in the **Control Flow Items** group in the **Toolbox**.

You can bring this task onto the **Control Flow** page by one of the following two methods:

4. Double click this in the **Toolbox**.

This task should appear on the **Control Flow** page, as shown in the following screenshot:

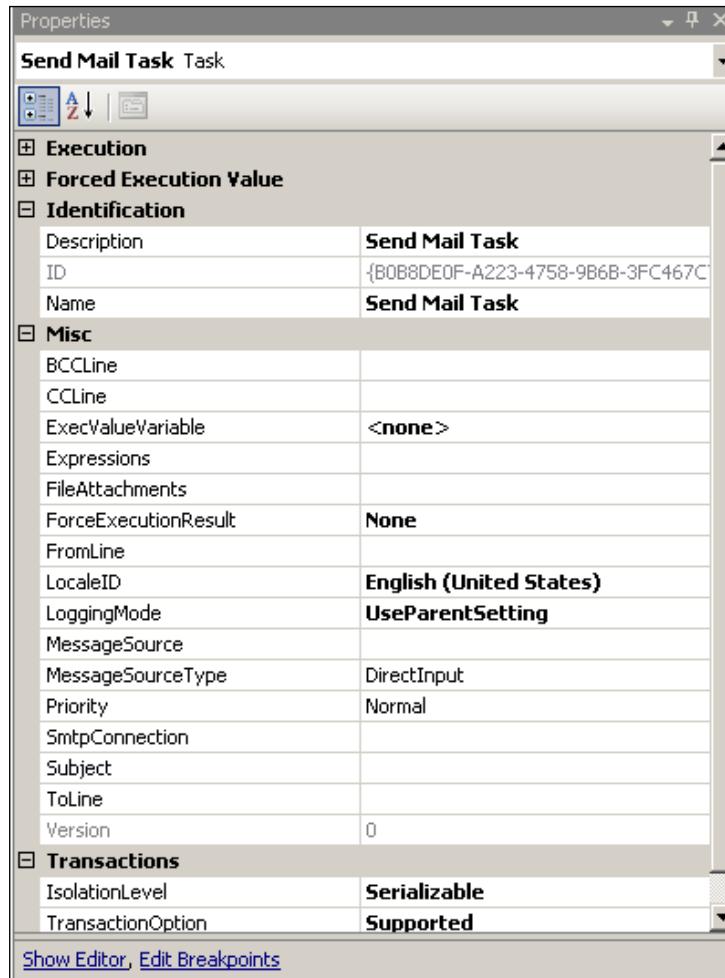


Or, you can click on the **Send Mail Task** in the **Toolbox** and drag it into the **Control Flow** page. In this case, the **Send Mail Task** will be placed where you release your mouse.

You notice a green arrow pointing downwards. This will be used if you want the control to flow to another object (at present left dangling). You also notice a white mark (x) inside a red dot in the **Send Mail Task**. If you place your mouse on top of it (hover over the mark) you get a pop-up that says that a SMTP server is not specified.

5. Click on the **Send Mail Task**.

Now you can access the properties window, as shown in the next screenshot.



You can edit the properties by clicking on the **Show Editor** link in this window. This will open the **Send Mail Task** editor which you will be opening by yet another method shortly

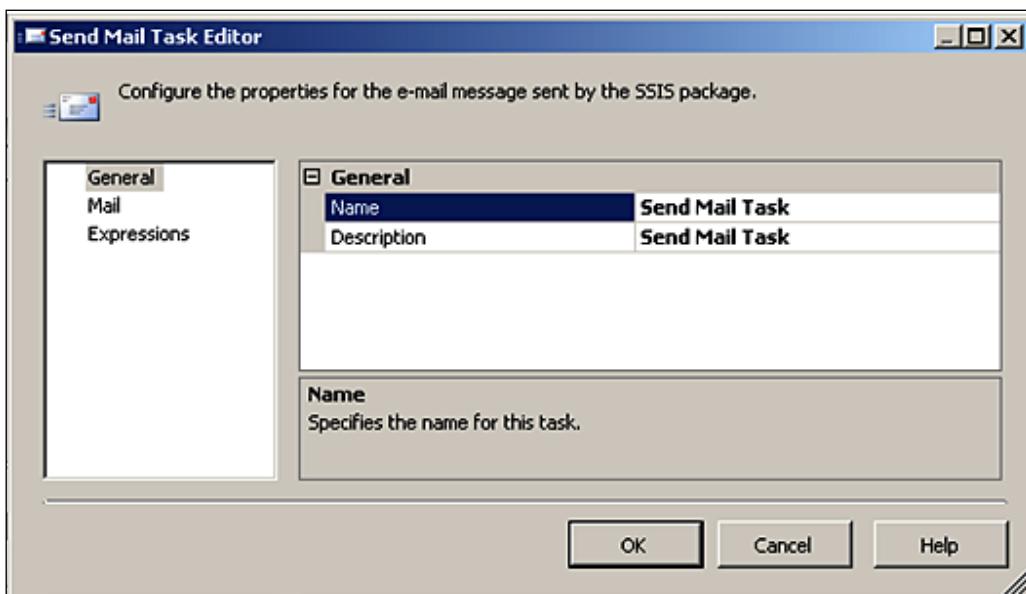
If you try to execute the package click on `Email.dtsx`, make a right click and choose **Execute Package**, you will get a **Package Validation Error** message. Click the **OK** button on the error message screen.

6. On the **Send Mail Task** in the **Control Flow** page, make a right-click to display the drop-down. Within this menu, click on the sub-menu item **Edit....**

This opens the **Send Mail Task Editor** window, as shown in the following screenshot, with the default branch, **General**.

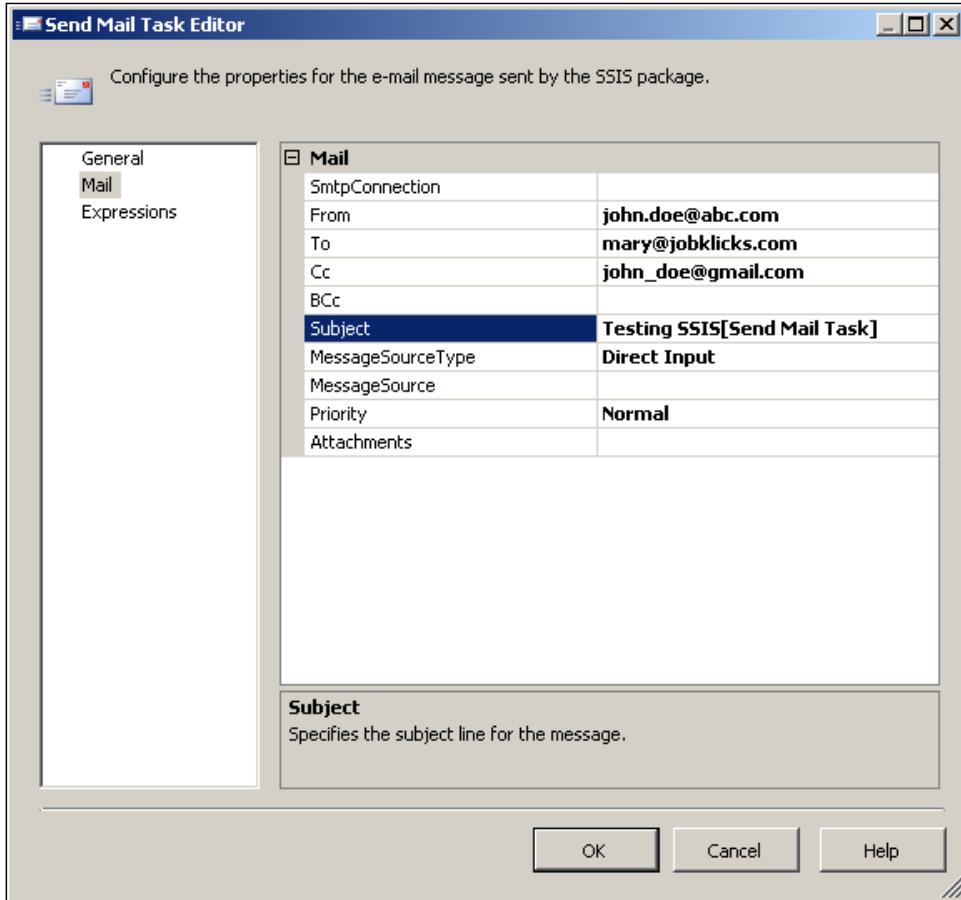
The same editor would have shown up had you clicked on the **Show Editor** hyperlink in the Send Mail Task's property window seen earlier.

This Editor has a list view in the left and the details of each of the list items on the right. Read the instructions on this editor (the editor is used to configure the properties of email messages sent by this package).



7. Edit the branch **General**. The **Name** and **Description** (both of which read, **Send Mail Task**) have been changed. Now the **Name** reads, **My Email Task** and the **Description** reads, **Send an email using my ISP's SMTP Server**. You may use your own **Name** and **Description**.

8. Now click on the branch **Mail**. This brings in the **Mail** window on the right side as shown in the following screenshot.



Here, there are a number of items, with some of them having options. You may notice, however, that a lot of them are familiar to you if you have used any of the email programs; such as: *FROM*, *TO*, *Cc*, *BCc*, *Subject*, *Priority*, *Attachments*, etc. The choices made are typed into the text boxes, as shown in the above screenshot. If needed, you can indicate attachments by clicking on the **Attachments** field, which opens up a window, **Open**, where you can navigate to locate the file you want as an attachment. For this exercise, the **Attachments** and the **BCc** fields are left blank.

9. Now click on an empty area along the line item **SMTP Connection**.

In the drop-down menu, click on **New Connection....** This brings up the window shown in the following screenshot.



10. Fill in the details, as shown in the next paragraph, and click on the **OK** button.

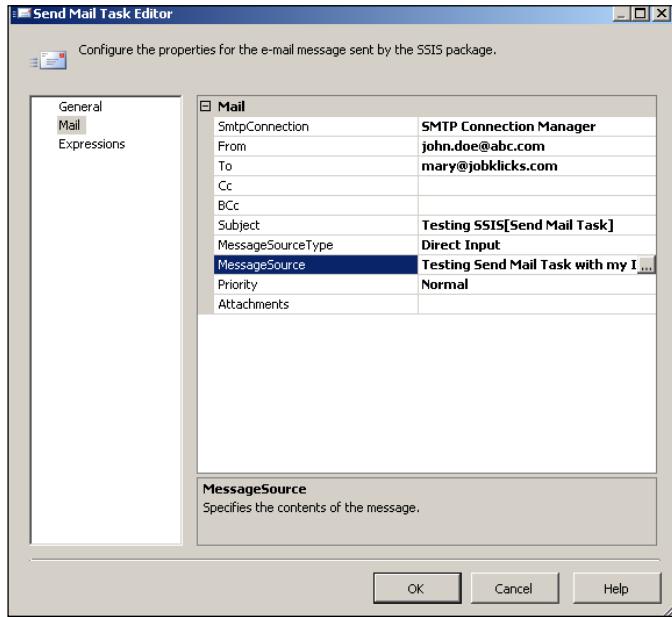
For **Description**, type in a suitable description of your choice such as "My ISP's SMTP Server" and in the **SMTP server** field type in the name of the SMTP Server provided by your ISP, like `SMTP.myISP.com`.

This enters the SMTP server information into the **Send Mail Task Editor**.

For the **MessageSourceType**, we have accepted the default, **Direct Input**.

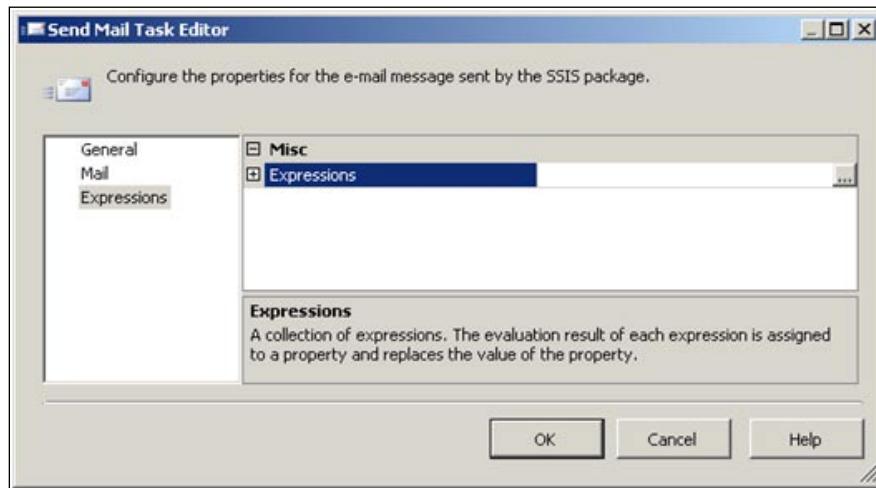
For the **MessageSource**, directly type-in the message **Testing the Send Mail Task with Direct Input and My ISP's SMTP Server** as shown in the window named "Message Source", or in the window (a small text editor window) that pops-up when you click on the ellipsis button in the **MessageSource** field.

This completes all the required fields, as shown in the following screenshot.



In this exercise, we used the default **Direct Input** for the **MessageSourceType** field in the above window. There are two other options: **File Connection** and **Variable**. If you were to choose **File Connection**, you will have to tell the wizard where the file is located. On the other hand, if you choose **Variable**, then the wizard will look for a variable you might have created to pull its value. The main objective in the present chapter is to describe the simplest of the many possible variations.

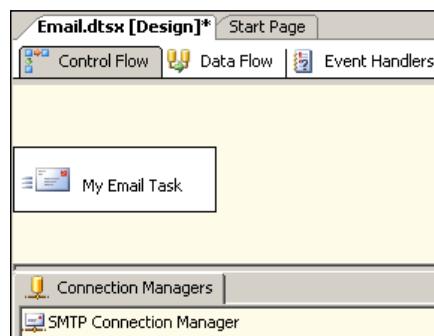
When you click on the **Expressions** list item in the left, it opens a window, as shown in the next screenshot. You may want to use an expression to evaluate a function of a type (by this, return value of function is meant) such as a string or a date (given today's date, a future date or a date in the past, for example) and use that calculated value in the task.



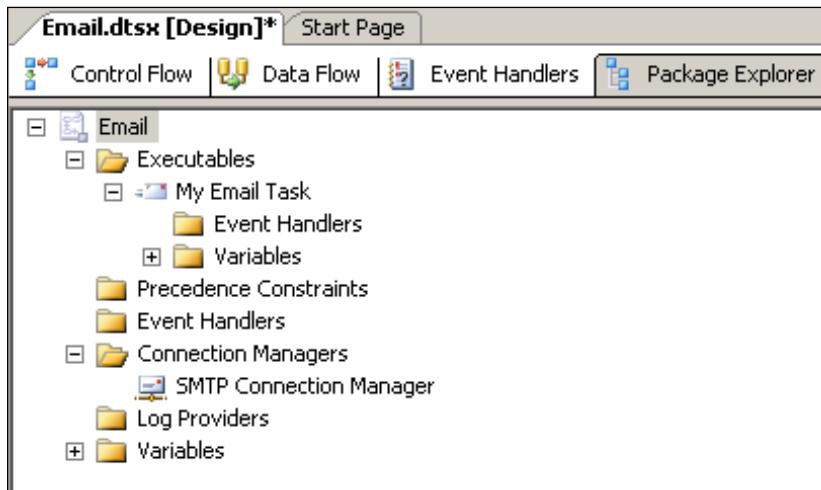
Here, the **Expressions** window is not configured. However, you can configure this window by associating a variable (that you define) to one of the allowed **Property Expressions** such as the *cc*, *bcc*, *subject* and other fields of the email that you are sending. In this exercise, to keep the explanation to bear minimum, **Expressions** list item is left empty.

11. Click on the **OK** button in the **Send Mail Task Editor**.

Review the changes made to the **Send Mail Task** property window. Also notice that the red dot with white cross has disappeared and a **Connection Manager** has been added to the **Connection Managers**' collection.



12. Click on the **Package Explorer** and verify that the **My Email Task** and the **Connection Manager** are included in the package.



13. Now click on the **Build** menu item and click on **Build Ch 3**.
You get a **Build Succeeded** message at the bottom-left of the screen.
14. Right-click the **Email.dtsx** in the **Solution Explorer** and click on the **Execute Package** drop-down menu item.

The program starts to run; the **Send Mail Task** (in the **Control Flow** page) first turns yellow and finally is filled with green color. Green is for success and red indicates failure. If want to return to the design mode, you should go to the menu item **Debug** and choose **Stop Debugging** from the drop-down menu.

15. Now, it is time to go to your email client program, such as Outlook Express or a similar program, to verify if the email message has left (browse the sent messages) or you may also use a web mail program such as Google or Yahoo, etc.

It may be noted that the Send Mail Task will not check if the account is correct. Also, you will be indicating whether or not you are using Windows authentication. Since the package is developed by the owner of the computer, the authentication was left blank. Even if this mode of authentication was implemented, the result would still be the same because the administrator of the computer created the package object.

Hands-On Exercise Two: How to Find Your ISP's SMTP Server?

On your desktop, you probably have email programs such Outlook Express, which is most common, and Microsoft Outlook. I have Outlook Express 6. You can find your ISP's SMTP server from the menu item **Tools** in Outlook Express, which displays a drop-down menu from which you choose **Accounts**. When you click on **Accounts**, it opens up the **Internet Accounts** window. On this window, choose the **MAIL** tab and click on the **Properties** button on the right of the window, after highlighting your **Account** in the above. It opens the properties for that account showing five tabs: **General, Servers, Connection, Security, and Advanced**.

You will find the SMTP server's correct name in here under **Server** tab, as shown in the next screenshot.



Summary

The Send Mail Task was chosen as it is the most ubiquitous program used . The Hands-On Exercise guides you through the process of configuring the task and executing the task. If you do not have an email program on your desktop (known as an email client) you may use the SMTP Server of your favorite web email, such as Google (<http://hodentek.blogspot.com/2007/06/configuring-web-emails-smtp-and-pop.html>) or Yahoo. Depending on the geographical region, you may choose an SMTP server from this comprehensive list of SMTP Servers (<http://www.e-eeasy.com/SMTPServerList.aspx>). You may also find some free (http://www.emailalarms.com/downloads/1st_smtp.html) SMTP servers on the Internet. If you are faced with connectivity problems for the SMTP server, you can use the Telnet program (http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/telnet_commands.mspx?mfr=true) and try to connect to port 25 or 587, if successful verifies connectivity.

4

Transferring Data to a Text File

This chapter shows you how to create a package that can transfer data from a table in an SQL 2005 Server database to a Flat File on the C:\ drive of your computer.

Flat file data storage predates relational databases. Every detail about a record consisting of a number of data fields is stored flat, in just one line in the file. Each data field is separated from the other by some kind of field separator, such as a tab, white space or a comma. There are no relationships between the fields in flat files. Flat files are sometimes needed to obtain a simple text-based version of data that can be transferred more easily between operating systems than data in a table. In the hands-on exercise, you will be transferring data retrieved from an SQL Server 2005 table to a flat file. You will be using a Data Flow task consisting of a source connected to an SQL Server 2005 based connection manager and a destination connected to a Flat File connection manager.

Hands-On Exercise: Transferring Data to a Text File

In order to follow the steps as indicated, you will need a source and a destination; the source data will be coming from SQL Server 2005, and the destination would be writing this to a flat file on your hard drive. You also need to establish a path connecting them.

Step 1: Creating a BI Project and Adding a Data Flow Task

1. Create a business intelligence project **Ch 4** as described in Chapter 2.
2. Change the default name of the package to **TableToText.dtsx**. Drag and drop a **Data Flow Task** from the **Toolbox** on to the **Control Flow** page.
3. Click the **Data Flow** tab that displays the **Data Flow** page.

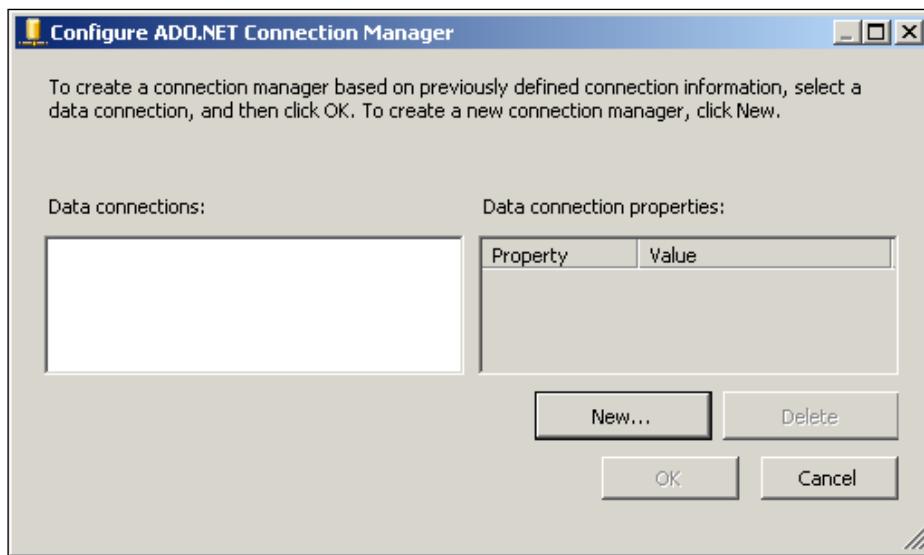
Now, you will be able to access the Data Flow items of the **Toolbox**, consisting of **Data Flow Sources**, **Data Flow Destinations** and **Data Flow Transformations** (refer to Chapter 1). Drag and Drop a **DataReader Source** from **Data Flow Sources** group onto the **Data Flow** page.

Step 2: Adding Connection Manager for the DataReader

The DataReader Source connects to the SQL Server 2005 using the connection properties defined in a connection manager.

1. Right-click inside the page of the **Connection Managers** (below the **DataFlow** page) and from the drop-down list of new connections, choose **New ADO.NET Connection....**

This opens the **Configure ADO.NET Connection Manager** wizard, as shown in the following screenshot.



If there is no configured connection manager, the **Data Connections:** area will be empty.

2. Click on the button, **New....**

This opens the **Connection Manager** window, shown in the following screenshot, where you will supply information required to get data from the SQL 2005 Server. All the fields (boxes) will be empty except for the **Provider:** field, which displays a **SqlClient Data Provider**.

3. Click on the drop-down handle for **Server name**.

SQL Servers on the computer should show up here. For a local server a "." may be used for the server name (127.0.0.1 is also acceptable). The local server is set for SQL Server authentication, which requires both **User name:** and **Password**.

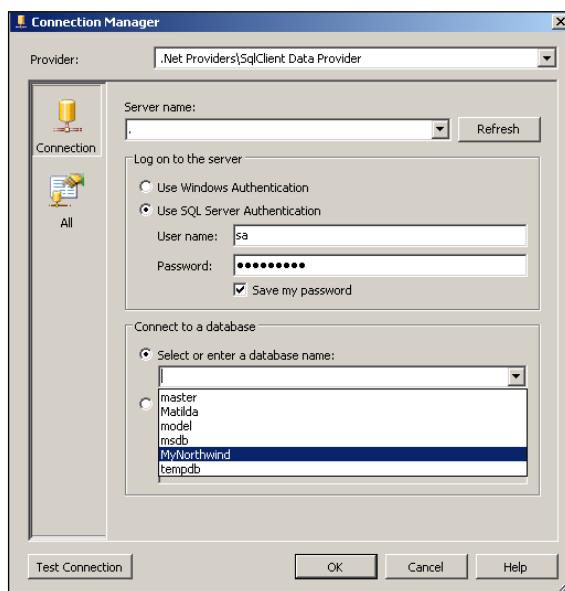
4. Type in a "." for the **Server name** as well as the **User name** and **Password** that you use for authentication in the respective fields.

If the information is correct, you will be able to display all the databases on this server.

5. Click on the drop-down handle for the field **Select or enter a database name:** in the **Connect to a database** area.

A list of all the available databases will be displayed as shown below.

SqlClient Data Provider is the provider that brings data fastest from an SQL Server 2005. You can test the connectivity using the **Test Connection** button, which will display a message saying the connection was successful.

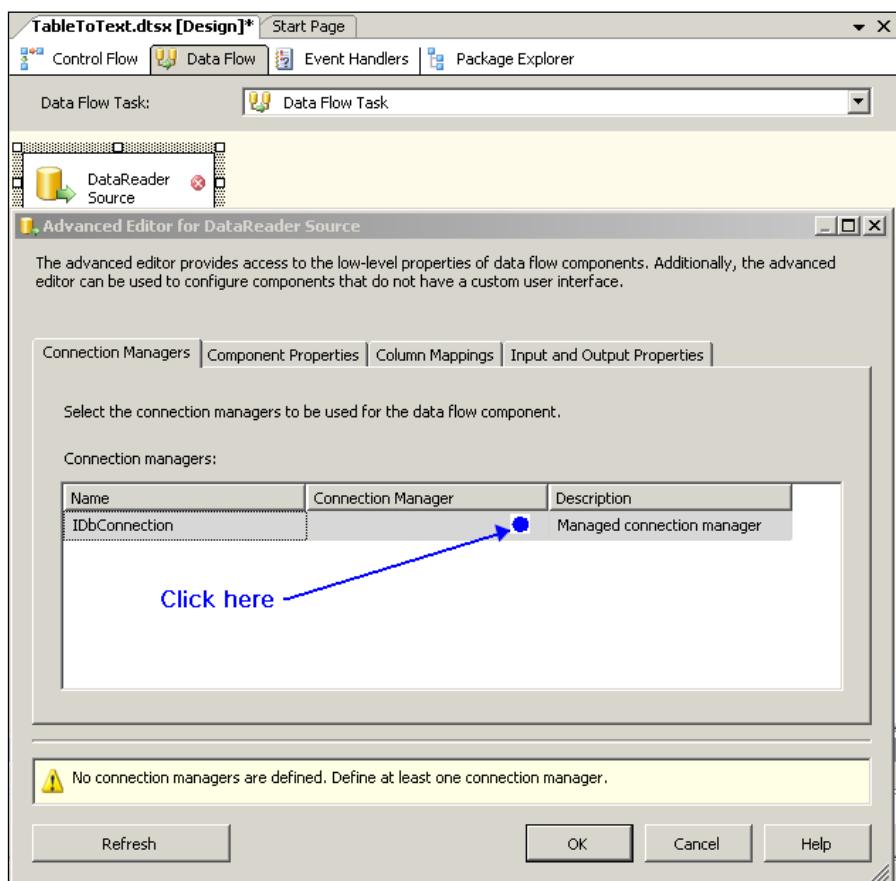


6. Click on the **OK** button on this screen.
7. Click on the **OK** button on the **Configure ADO.NET Connection Manager** window, shown in the previous screenshot.

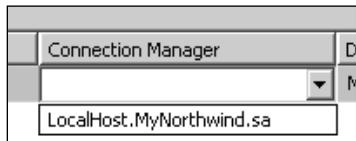
A connection manager, **localhost.MyNorthwind.sa**, will be added to the **Connection Manager's** page.

Step 3: Configuring the Source

Right-click the **DataReader Source**, in the drop-down menu, choose **Edit....** This opens the **Advanced Editor for DataReader Source**, as shown below. First, you need to indicate a connection manager that the **DataReader** can use.

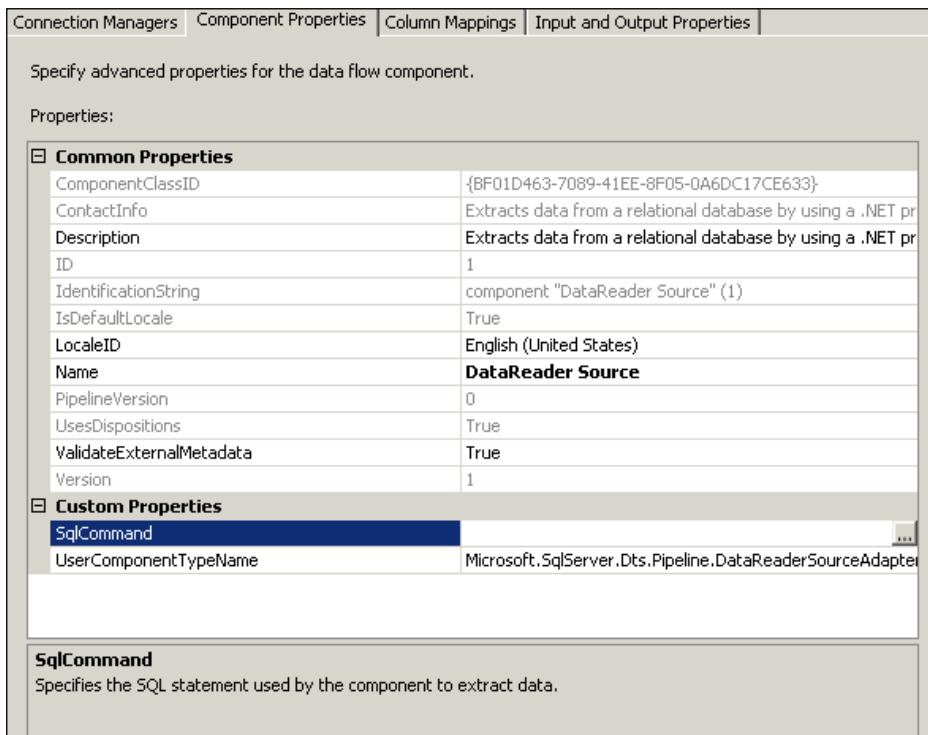


1. Click on the location (shown in the previous screenshot) and from the drop-down list, choose the connection manager you configured in Step 2, displayed as shown in the following screenshot.
2. Select this connection manager by clicking on this item.



3. Next, click on the **Component Properties** tab to open the properties of the DataReader component.

Here you will notice that this requires an **SqlCommand** (empty for now), as shown in the following screenshot.



4. In the **SqlCommand** field, click on the ellipsis button along its side.

You will pop open a text editor where you can type in your **SqlCommand**. You may also directly type-in the **SqlCommand** chosen for the exercise in this field, as in the next mini-step.

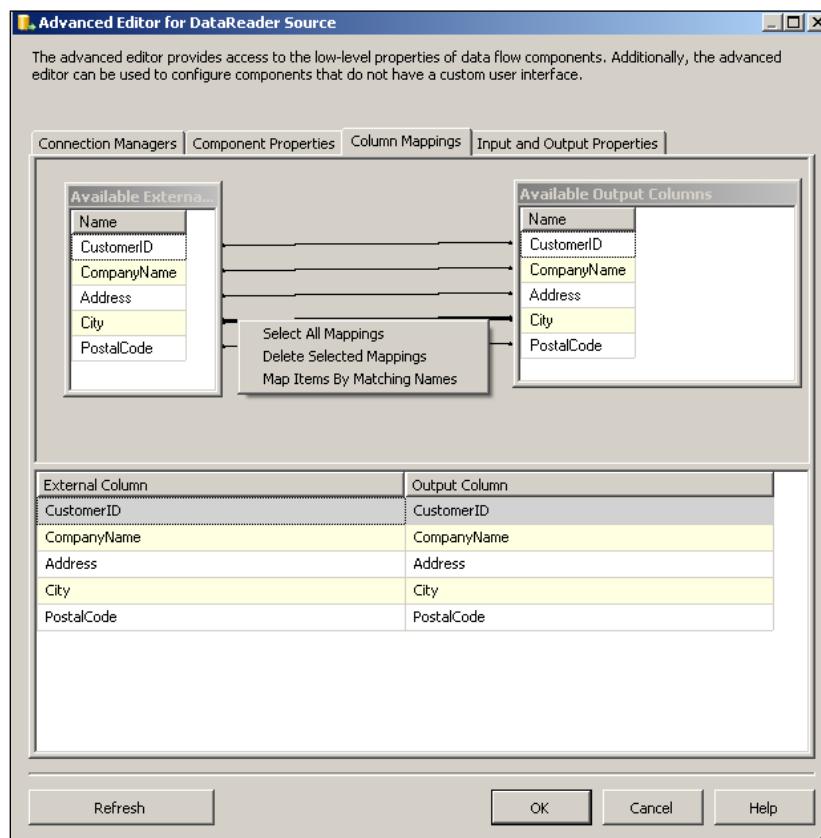
5. Type in "**Select CustomerID, CompanyName, Address, City, PostalCode from Customers**".
6. Before you move onto the next tab, click on the **Refresh** button.

[ You may create the text file using a File System Task (we'll be discussing that in the later chapters)]

7. Click on the **Column Mappings** tab.

This will open the **Column Mappings** page, showing the columns that are the available outputs from the DataReader.

The column mappings shown connecting the available external columns to the available output columns can be independently modified, although no such modification will be made. This is just to illustrate the flexibility you have in modifying column mappings. You can modify, individually, by clicking on the connection you want to modify, and then making a right-click to display the drop-down shown in the inner window.



In the last tab on this editor, **Input and Output properties**, you can add/ remove items from the **External Columns**, the **Output Columns**, and the **DataReader Error output**. For this exercise no modifications are made.

8. Click on the **OK** button in the above window.

This completes the configuration of the DataReader that brings five columns from the SQL 2005 Server.

Step 4: Adding a Flat File Destination and Establishing a Path from DataReader Source

In this step, we will add a Flat File destination component to the **Data Flow** page.

1. From the **Toolbox** menu, drag and drop the **Flat File Destination** component, under the **Data Flow Destinations** group, onto the **Data Flow** page.
2. Now right-click **DataReader Source**, and from the drop-down menu choose **Add Path**.

This opens up the window, **Data Flow**, which allows you to establish a data flow path and displays the **From:** location as **DataReader Source**.

3. Click on the drop-down **To:**, which shows both the **DataReader Source** as well as the **Flat File Destination**.
4. Choose the **Flat File Destination** and click the **OK** button.

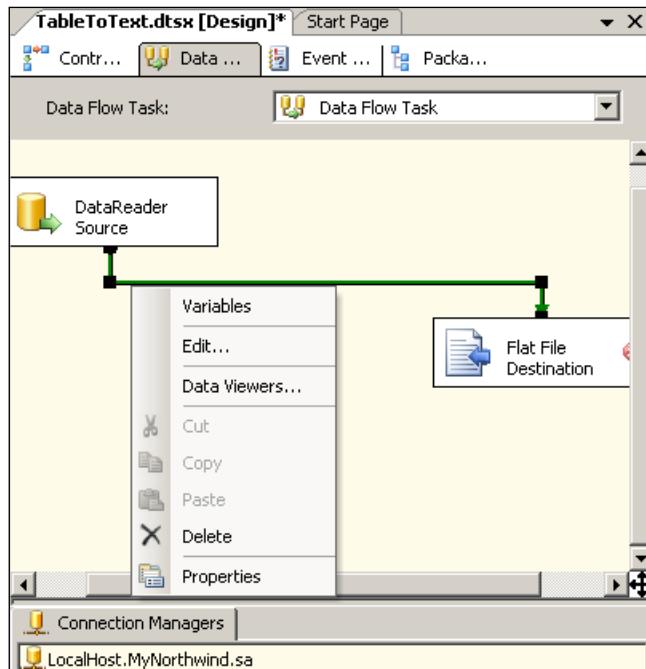
This opens up the **Input Output Selection** window, which shows the **output** and **input** windows available. You need to choose the appropriate ones. The path should connect from the **DataReader Output** to the **Flat File Destination Input**.

5. Click and choose the **Flat File Destination**.
6. Now click on the **OK** button in the **Data Flow** window.

You will see a green line connecting the **DataReader Source** to the **Flat File Destination**, as shown in the following screenshot.

Alternately, the process of establishing the path can be simplified by just picking the green dangling line from **DataReader** with the mouse, and extending it to touch the **Flat File Destination** object on the Data Flow page.

As seen in the following screenshot, you may edit the path after it is created by right-clicking this green line and choosing **Edit...** from the drop-down menu.



Step 5: Configuring the Flat File Destination Component

The data is on its way through the "path" represented by the green line in the previous step. The **Flat File Destination** also, requires a connection manager.

The **Flat File Destination** connects to a text file on your hard drive using the connection properties defined in a connection manager.

1. Create a text file, `Chapter4.txt`, (for this exercise using *Notepad*) in the `C:\` drive.
At present, the `Chapter4.txt` is empty.
2. Go ahead and type in a line of text for the column headers separated by commas, as shown in the next paragraph:
`CustomerID,CompanyName,Address,City,PostalCode`
3. Right-click a clear area inside the Connection Managers' page and from the drop-down choose, **New Flat File Connection....**

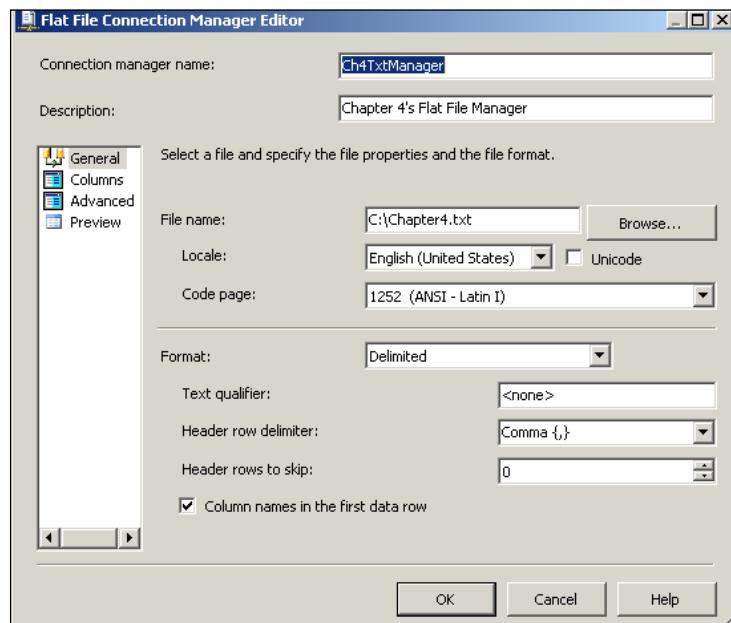
This opens the **Flat File Connection Manager Editor**, as shown in the following screenshot.

To start with, all the fields in the **Flat File Connection Manager Editor** window are empty.

4. Type in a name for the connection manager and an appropriate description of your choice. The name chosen for the connection manager is "Ch4TxtManager" and the description, "Chapter4's Flat File Manager".
5. Now click on the button **Browse...** to locate the Chapter4.txt created in the C:\ drive after navigating to the C: drive.

If you have not created a text file to start with, in the **Open** window for locating the file, you can create the file, and immediately click on the created file (but then you have to add the Column headers as shown previously). This gets added with default file format information (**Locale**, **CodePage**, **Head Row Limiter** and **Head rows to skip**), as shown in the following screenshot.

6. Also place a check mark in the **Column names in the first data row**, as we chose to add Column names.
7. Choose items related to **Format** the same way the text file was formatted in mini-step 26 (comma delimited, headers rows not skipped).
8. When you click on the **Columns** in the left-hand side, you see displayed the column names you created earlier in the Chapter 4.txt file.



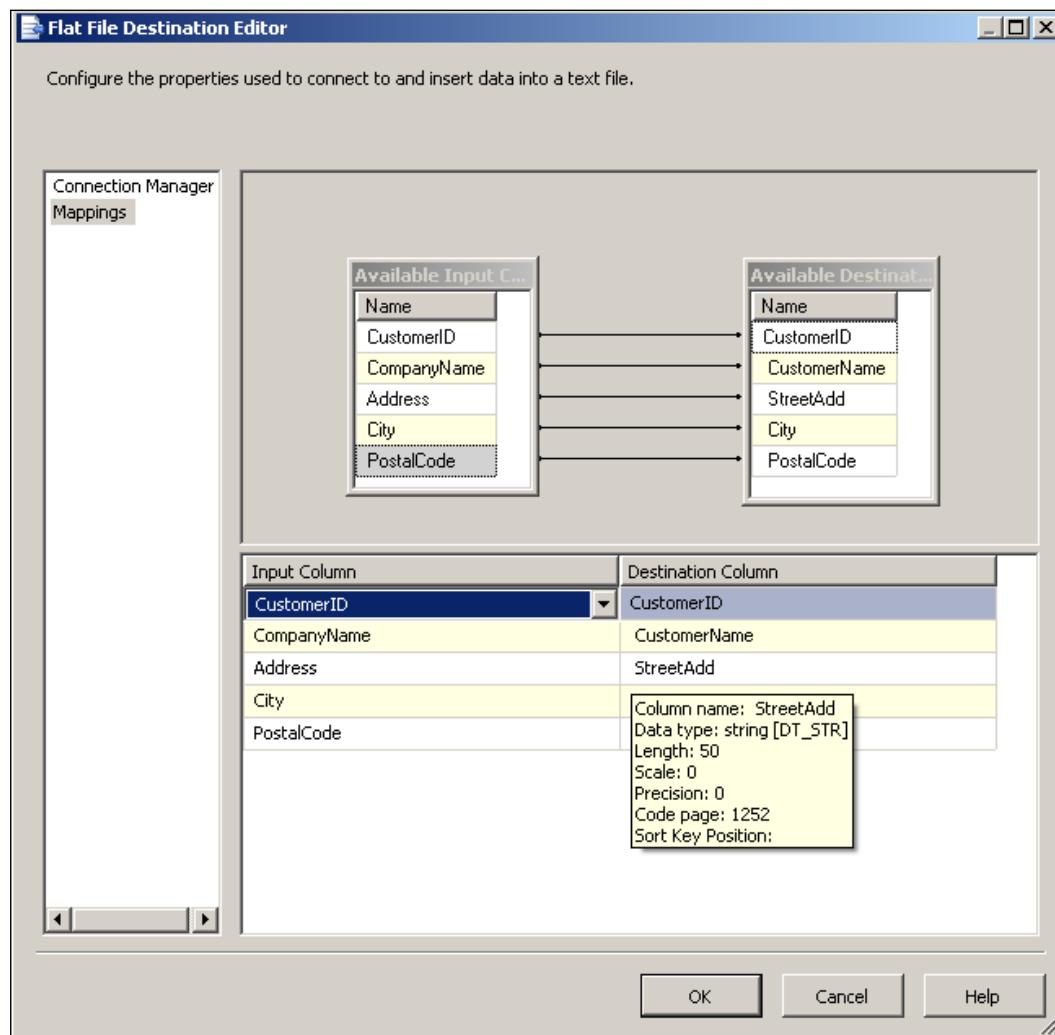
9. Click on the **OK** button on the **Flat File Connection Manager Editor**.

This adds the connection manager, **Ch4TxtManager** to the **Connections Managers**' page in the Canvas.

10. Right-click the **Flat File Destination** component and choose **Edit....**

This opens the **Flat File Destination Editor**, displaying the recently configured connection manager (it is also possible to configure a new connection manager starting from here as well). In Microsoft programs, there is more than one way of carrying out a task.

11. Click on the **Mappings** list item on the left side of this window, as shown in the following screenshot.



On the right-hand side, you see how the columns are mapped to the destination columns. Here, you see the association between the columns that are coming into the Flat File and those that are being written to the destination.

12. Place your mouse on any of the columns (**Input Column** or **Destination Column**).

You will see a tool-tip text window displaying the information for that particular columns, as shown for **StreetAdd** column.

13. Click on the **OK** button in this screen and you are done.

Step 6: Build and Execute the Package

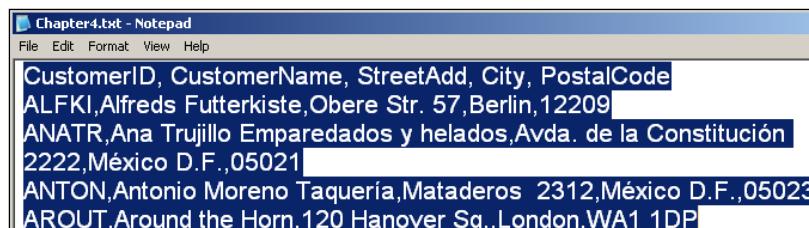
1. Yes, click on the **Build** menu item.
2. Now right-click the package file `TableToText.dtsx` in the **Solution Explorer**, and from the drop-down choose **Execute Package**.

The program starts up and after a while the data flow components change to green, after turning briefly to yellow at first (red means failure). You will also see that 92 rows were transferred (you will see "92 rows" by the side of the green path line connecting the source and the destination).

You will see messages in the Output window with the final line displaying successful execution of package.

3. Also click on the **Progress** tab, which now appears in the Canvas by the side of the **Package Explorer** to review how the execution started and completed. If the package failed, this will also show the step at which the package failed.
4. Finally open up your `Chapter4.txt` file and verify that the table data is transferred to the text file.

Only a part of the (four rows) data from this text file is shown in the following screenshot.



Summary

This chapter described the configuration of data flow components that are necessary for transferring data from a single table on the SQL Server 2005 to a pre-formatted Flat File. The mappings between the columns of table to the text file must be correctly carried out for successful transfer. The reverse of this transfer can be accomplished by choosing a Flat File Source and an SQL Server Destination.

5

Transferring Data to a Microsoft Excel File

This chapter shows you how to create a package that can transfer data from a table in an SQL Server 2005 Server database to a Microsoft Excel Spreadsheet. You will also learn how to use a Character Map Data Transformation. In the hands-on exercise, you will be transferring data retrieved from an SQL 2005 table to MS Excel 2003 spreadsheet file. You will be using a Data Flow Task consisting of a source connected to a SQL 2005 Server-based connection manager, as in the previous chapter, and an Excel Destination connected to an Excel connection manager.

Hands-On Exercise: Transferring Data to an Excel File

This exercise consists of the following major steps:

- Creating a BI project and adding a Data Flow Task.
- Configuring the DataReader's Connection Manager.
- Configuring the DataReader Source.
- Adding a Character Map Transformation.
- Adding an Excel Destination and establishing a path to it from the Character Map Data Flow Component.
- Configuring the Excel Destination Component.
- Testing the package.

In order to follow the steps as indicated, you will need a source and a destination: the source data is extracted from the *MyNorthwind* database (just a renamed version of the *Northwind* database) on the SQL Server 2005 Server, and the destination is loading this to an MS Excel 2003 spreadsheet file on your hard drive. You also need to establish a path connecting them. In addition, you will also interpose a Character Map Data Flow Task that will convert the text in one of the data fields, so that all characters in that column are capitalized after the transformation.

Step 1: Creating a BI Project and Adding a Data Flow Task

In this section, you will be creating a Business Intelligence project and changing the name of the default package object. Since it is data-related, you will be adding a Data Flow Task. You will also be adding a DataReader component to the data flow.

1. Create a business intelligence project **Ch 5** as described in Chapter 2 or Chapter 3.
2. Change the default name of package from `Package.dtsx` to `TableToXls.dtsx`.
3. Drag and drop a **Data Flow Task** from the **Toolbox** onto the **Control Flow** page.
4. Click open the **Data Flow** tab, which displays the **Data Flow** page.
Now, you will be able to access the Data Flow Items of the **Toolbox** consisting of **Data Flow Sources**, **Data Flow Destinations**, and **Data Flow Transformations** (refer to Chapter 1).
5. Drag and drop a **DataReader Source** from the **Data Flow Sources** group onto the **Data Flow** page.

Step 2: Configuring the DataReader's Connection Manager

Configuring the **DataReader** source that connects to the local SQL Server 2005 has been described in earlier chapters. Here, only a couple of the images relevant to this chapter will be shown.

1. Right-click inside the **Connection Managers** page below the Canvas, and from the drop-down choose **New ADO.Net Connection....**

If you are continuing with this chapter after Chapter 4, you will see the **Configure ADO.NET Connection Manager** screen displaying the previously configured connection manager. If you need to create a new one, follow the steps shown in the previous chapter.

2. Click on the **OK** button in the **Configure ADO.NET Connection Manager** window.

A connection manager, **localhost.MyNorthwind.sa**, will be added to the **Connection Manager's** page.

Step 3: Configuring the DataReader Source

1. Right-click the **DataReader Source**, in the drop-down menu.
2. Choose the **Edit...** menu item in the drop-down menu.
This opens the **Advanced Editor for DataReader Source**. At first, you need to indicate a connection manager that the DataReader can use.
3. In the **Advanced Editor for DataReader Source** that gets displayed click on the **Connection Managers** tab.
4. Click on an empty area (in grey) below the list heading, **Connection Manager**.
Here, you will see the connection manager that you added in step 1.
5. Choose this **Connection Manager**.
6. Next, click on the **Component Properties** tab to open the properties of the DataReader component.

Here, you will notice that this requires an **SQLCommand** (the only empty field now).

7. Click on the ellipsis button along its side to display a text editor where you can type in your **SQLCommand**.

You may also directly type-in the SQL Command:

```
SELECT CustomerID, CompanyName, Address, City, PostalCode  
FROM Customers
```

8. Click on the **Refresh** button.

This query will allow the DataReader to read the data from the five columns. A sample of the table data is shown in the following screenshot, taken from SQL Server 2005 Management Studio. If you recall, these were the same columns that were used in the previous chapter as well.



The screenshot shows a SQL Server Management Studio window with a query editor titled '(local).MyNort...SQLQuery1.sql*'. The query is: 'Select CustomerID, CompanyName, Address, City, PostalCode from Customers'. Below the query is a results grid with the following data:

	CustomerID	CompanyName	Address	City	PostalCode
1	ALFKI	Alfreds Futterkiste	Obere Str. 57	Berlin	12209
2	ANATR	Ana Trujillo Emparedados y ...	Avda. de la Constitución ...	México D.F.	05021
3	ANTON	Antonio Moreno Taquería	Mataderos 2312	México D.F.	05023
4	AROUT	Around the Horn	120 Hanover Sq.	London	WA1 1DP

9. Click on the **Column Mappings** tab.

This will open the **Column Mappings** page showing the columns that are the output of the DataReader.

In the last tab on this editor, **Input and Output properties**, you can add/ remove items from the **External Columns**, the **Output Columns** and the **DataReader Error output**. For this tutorial, no modifications are made.

10. Click on the **OK** button in the above window.

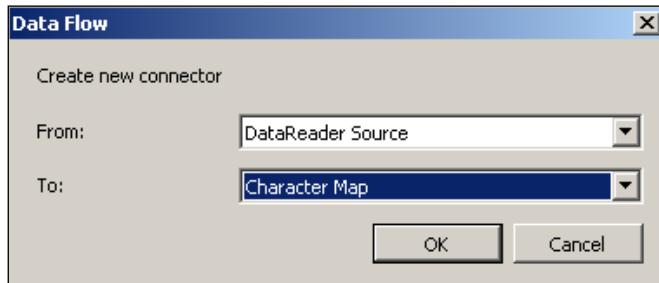
This completes the configuration of the DataReader which brings five columns from the SQL 2005 Server.

Step 4: Adding a Character Map Transformation

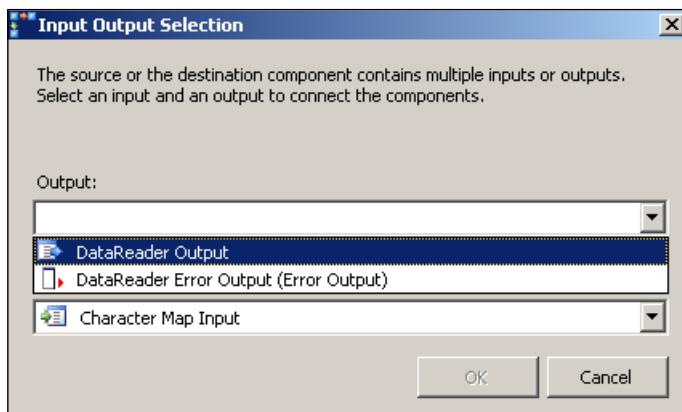
The *Character Map* transformation is described in Chapter 1, but here you will be experimenting with this transformation. The transformation manipulates the text string that is coming to it and outputs the manipulated string. For example, in the screenshot we have just seen above, the **CompanyName** has mixed case. Using this transformation, we will capitalize all the characters that appear in the **CompanyName** column before it is written to an Excel File – Alfreds Futterkiste will become ALFREDS FUTTERKISTE, etc.

1. Drag and drop a **Character Map** data flow item from the **Data Flow Transformations Group** in the **Toolbox** onto the **Data Flow** page of the Canvas.

2. Right-click on the **DataReader Source** and from the drop-down click on the **Add Path** menu item.
3. From the displayed window, **Data Flow**, choose **Character Map** for the **To:** field as shown.



4. Click **OK** in the above window, and the following window is displayed.



Here, you need to indicate the output from this component, from the drop-down shown.

5. Choose **DataReader Output Source** from the drop-down.

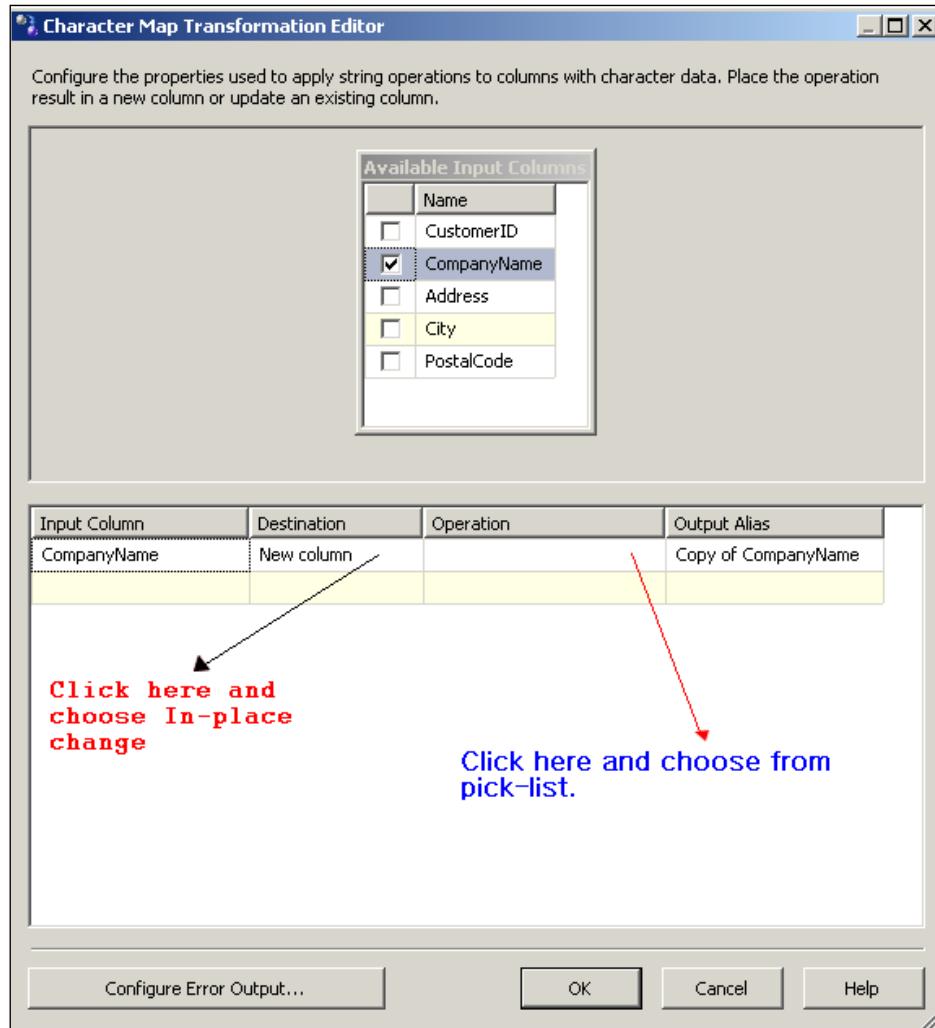
At present, you will not be dealing with any errors in this tutorial. When you choose the above option, the **OK** button gets enabled.

6. Click on the **OK** button in the above window.

This establishes the path from the **DataReader Source** to the **Character Map** data flow component. The path is established but it still needs configuration.

7. Right-click the **Character Map** component and from the drop-down menu choose, **Edit**.

This opens the **Character Map Editor**, as shown in the following screenshot.
Place a check mark for the **Company Name** column.

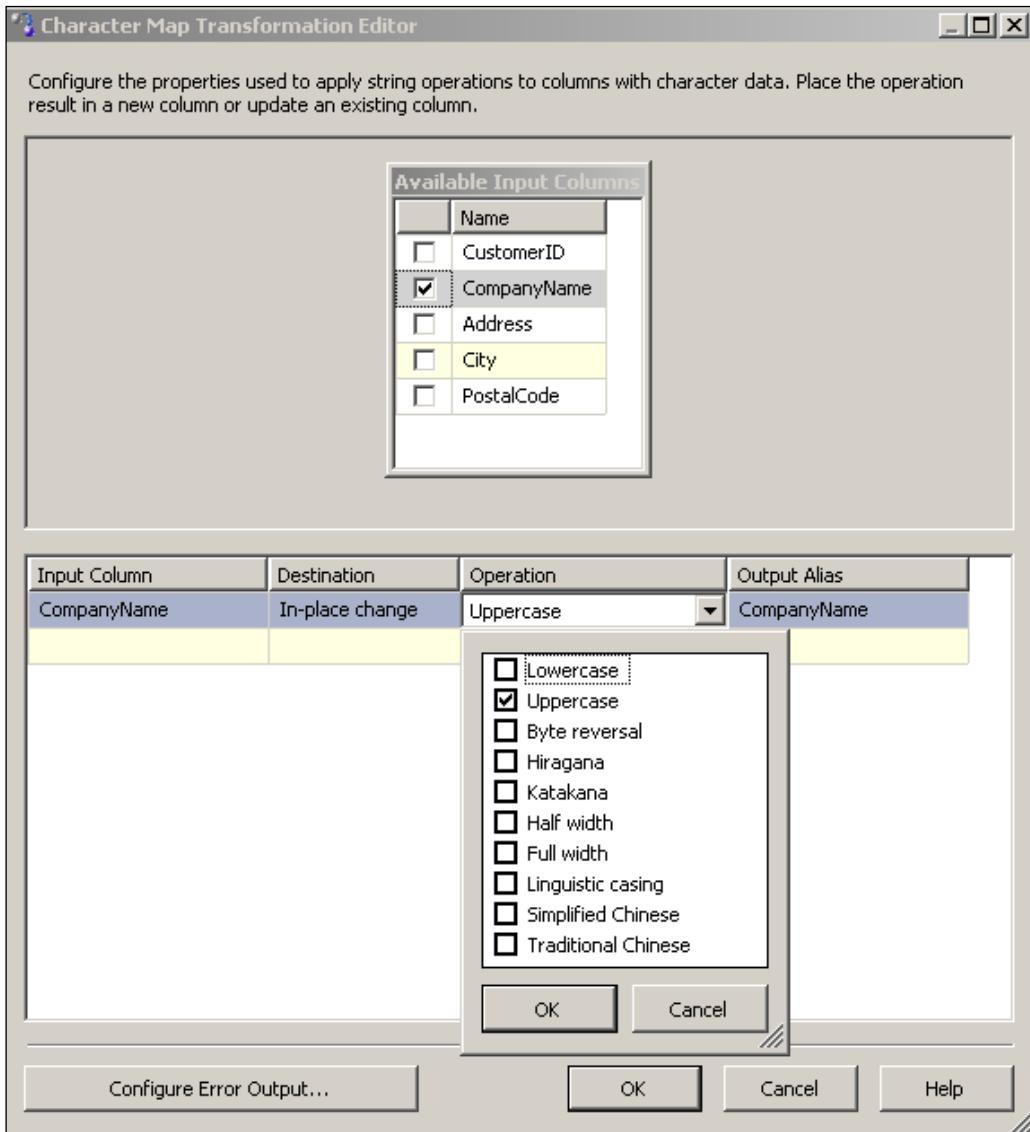


If the default as shown in the above were to be chosen, then an extra column will be added to the output. We choose the option **In-place change** from the drop-down.

8. Click on the cell, **New Column**, under **Destination** in the above window. From the drop-down choose, **In-place change**.

9. Click just below the **Operation** list-header and from the drop-down list choose **Upper Case** as shown in the following screenshot.

The output alias remains the same as it is an in-line change.



10. Click on the button **OK** in the pick-up list and to the **OK** button in the **Character Map Transformation Editor**.

This completes the Character Map configuration.

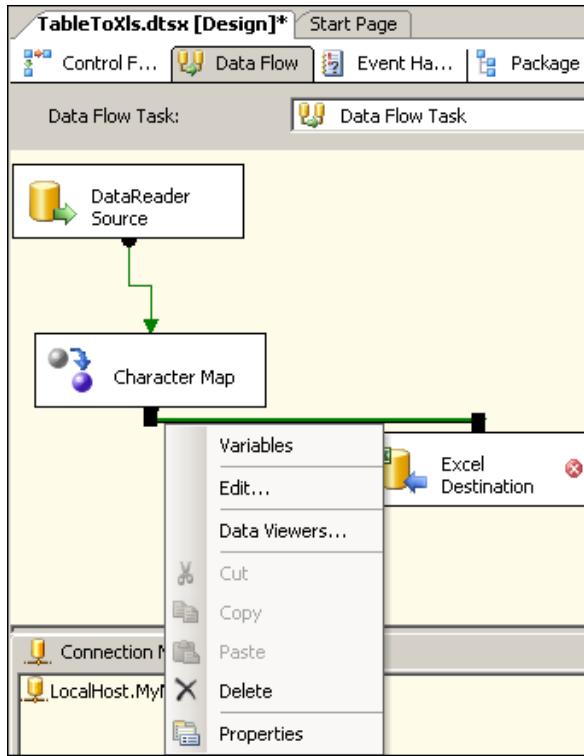
Step 5: Adding an Excel Destination and Establishing a Path to It from the Character Map Data Flow Component

In this step, we will add an *Excel Destination*. We will then establish a path from the *Character Map* to *Excel Destination*.

1. Add an **Excel Destination** component from the **Data Flow Destinations** group in the toolbox to the **Data Flow** page.
This can be accomplished either by double-clicking the component in the **Toolbox** or a drag-and-drop operation.
2. Right-click **Character Map** and from the drop-down menu choose **Add Path**.
This opens up the window, **Data Flow**, which allows you to establish a data flow path, and displays the "From:" location as **Character Map**.
3. Click on the drop-down along "To:", which shows both the **Excel Destination** as well as the **DataReader Source**.
4. Choose the **Excel Destination** and click **OK** to the screen.
This opens up the **Input Output Selection** window that shows the available **output** and **input** windows. The Output window is empty whereas the input shows **Excel Destination Input**. The path should connect from the **Character Map** to the **Excel Destination Input**.

5. Choose the above options and click on the **OK** button on this screen.

You will see a green line connecting the **Character Map** Data Flow Component to the **Excel Destination**, as shown in the next screenshot.
Alternately, the process of establishing the path can be simplified by just picking the green dangling line from **Character Map** and dropping it onto the **Excel Destination** object on the **Data Flow** page. As seen in the next screenshot, you may also edit the path after it is created by right-clicking on this green line and choosing the **Edit...** drop-down menu item.



Step 6: Configuring the Excel Destination Component

The data is on its way through the path, represented by the green line in the previous step. The Excel Destination also requires a connection manager.

The **Excel Destination** connects to an MS Excel on your hard drive using the connection properties defined in a connection manager.

1. Right-click the **Excel Destination** and from the drop-down menu choose **Edit**.

This displays the **Excel Destination Editor**. Excel requires an OLE DB connection manager and if there are no configured connection managers (by you or a previous user) the drop-down will be empty.

2. Click on the **New...** button.

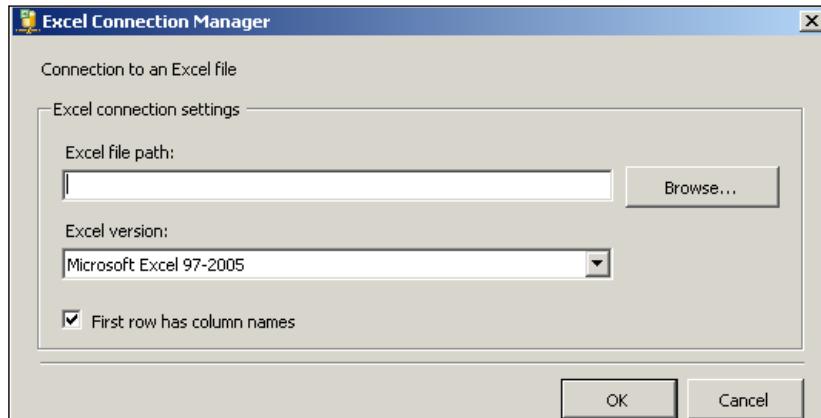
The **Excel Connection Manager** window gets displayed as shown in the next screenshot. Here, you need to use the **Browse** button and pick the Excel file as the destination. The data will be written to the destination when the package is run.

3. Open Windows Explorer and create an Excel file in the C:\ drive.

For this tutorial, TableToXls.xls is chosen.

Besides connecting to an existing file, the Excel Connection Manager supports creating a file on the folder of your choice in the machine using the **Browse...** button.

4. Now browse to the newly created file using the **Browse...** button and choose this file.
5. Click on the **OK** button in the **Excel Connection Manager** window.

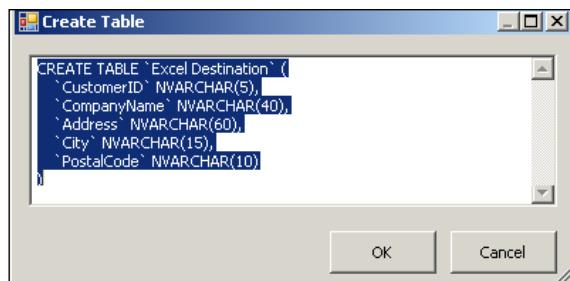


6. For the **Data Access Mode**, accept the default, **Table or View**.

You have to indicate the name of the Excel sheet that will be used. (Do not click on the drop-down for locating the sheet. The drop-down will show the three Excel sheets that are found in a newly created Excel worksheet file; all of them having just one column each.)

7. Click on the **New...** button.

You are creating a new Excel sheet. This pops-up a **Create Table** window showing the columns that are being piped into the component, as shown in the following screenshot.

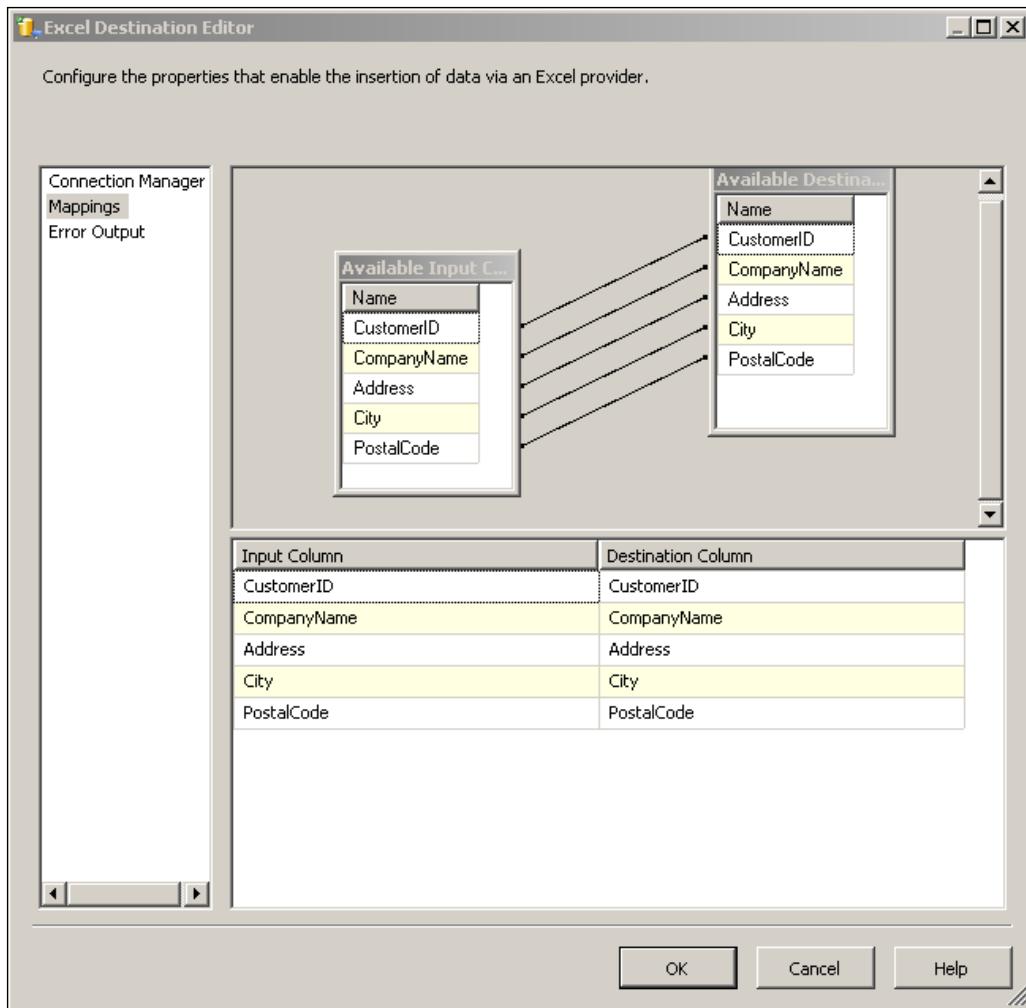


8. Click on the **OK** button, shown in the previous screenshot.

A new Excel Sheet, **Excel Destination**, will be added to the **TableToXLS.xls** file. If you now open and review this file (**TableToXls.xls**) you will see the column headers are added to this sheet.

9. Click on the **Mappings** in the left-hand-side pane of the **Excel Destination Editor**, which shows the mappings from the input to the output.

This shows all the columns from the **Character Map** Data Flow Component being written to the destination file, as shown in the following screenshot.



10. Click on the **OK** button in this window. The package is now completely configured and ready for execution.

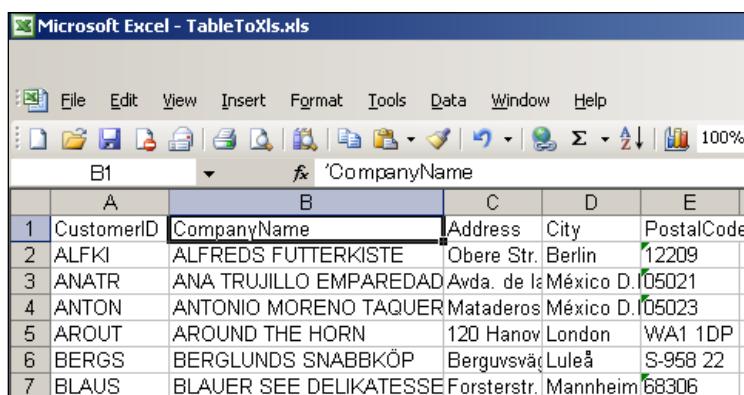
Step 7: Testing the Package

1. Right-click the TableToXls.dtsx in the **Solution Explorer** and from the drop-down choose, **Execute Package**.

All three components on the Data Flow page turn green, indicating that the package executed successfully without errors. You may review the **Progress** tab in the 'Canvas', which shows all the details of the execution of the package.

2. Now open up the TableToXls.xls file and review.

A few rows of data are shown in the next screenshot. Notice that the *Character Map Data Transformation* component has capitalized all the characters in the **CompanyName** column.



	A	B	C	D	E
1	CustomerID	CompanyName	Address	City	PostalCode
2	ALFKI	ALFREDS FUTTERKISTE	Obere Str. 57	Berlin	12209
3	ANATR	ANATRICAL EMPAREJAD	Avda. de la México 24	México D.F.	105021
4	ANTON	ANTONIO MORENO TAQUERIA	Mataderos	México D.F.	105023
5	AROUT	AROUND THE HORN	120 Hanov	London	WA1 1DP
6	BERGS	BERGLUNDS SNABBKÖP	Berguvsväg 8	Luleå	S-958 22
7	BLAUS	BLAUER SEE DELIKATESSE	Forsterstr.	Mannheim	68306

Summary

This chapter described the following:

- Configuring data flow components that are necessary for transferring data from single table on the SQL Server 2005 to an Excel file.
- The usage of a Character Map Data Transformation component.

The reverse of this transfer can be accomplished by choosing an Excel Source and an SQL Server Destination.

6

Data Transfer to an MS Access Database

This chapter shows you how to create a package that can transfer data from an SQL Server 2005 database to a Microsoft Access Database. You will also learn how to use a Data Viewer to monitor the data flow. Recently, Microsoft released the 2007 version, but this chapter will be using the 2003 version of this product.

In the Hands-On Exercise, you will be transferring data retrieved from an SQL Server 2005 table to an MS Access database's table. You will be using a Data Flow Task consisting of a source connected to an SQL Server 2005-based connection manager, as in the previous chapter, and a MS Access database accessed via an OLE DB Destination. You will also learn the usage of a Data Viewer for monitoring the data flow.

Hands-On Exercise: Transferring Data to an Access Database

In order to execute the steps as indicated, you will need a source and a destination; the source data is extracted from SQL Server 2005 and the destination is an MS Access Database table. You also need to establish a path connecting them. In addition, you will also incorporate a Data Viewer that will monitor the flow from the source to the destination.

The following are the major steps to be taken in this exercise:

- Creating a BI project and adding a Data Flow Task.
- Configuring the DataReader's Connection Manager.
- Configuring the DataReader Source.

- Adding an OLE DB Destination and establishing a path from the DataReader Component.
- Configuring the OLE DB Destination Component.
- Incorporating a Data Viewer to monitor data flow.

Step 1: Creating a BI Project and Adding a Data Flow Task

1. Create a business intelligence project **Ch 6** as described in Chapter 2 or Chapter 3.
2. Change the default name of the package to `TableToAccess.dtsx`.
3. Drag and Drop a **DataReader Source** from **Data Flow Sources** group onto the **Data Flow** page.

Step 2: Configuring the DataReader's Connection Manager

Configuring the connection manager for a DataReader source that connects to the local SQL Server 2005 has been described in detail in the previous chapter (steps 2 and 3 of Chapter 4). After this step, you should be seeing a connection manager `localhost.MyNorthwind.sa` in the **Connection Managers** page.

Step 3: Configuring the DataReader Source

1. Right-click the **DataReader Source** and choose **Edit...** from the drop-down menu.
Advanced Editor for DataReader Source, the connection manager gets displayed.
2. In the **Advanced Editor for DataReader Source**, click on the **Connection Manager** tab and choose `localhost.MyNorthwind.sa`, created in the previous chapter.
3. In the **Component Properties** tab, type in the following SQL Command into the textbox corresponding to the **SQLCommand** (which comes up empty at first) or type the statement in the String Value Editor that pops-up when you click on an empty area along the line item, **SQLCommand** under Custom Properties node in the Component Properties tab of the editor.

```
"Select [ProductName], [QuantityPerUnit], [UnitPrice],  
[ReorderLevel] from Products"
```



Caution: You cannot directly copy and paste the SQL statement from the SQL Server as each column needs to be enclosed by the square braces.

This query will allow the DataReader to read the data from the four columns. A sample of the table data is shown in the following screenshot, taken from SQL Server 2005's Management Studio.

The screenshot shows a SQL query window titled '(local).MyNort...SQLQuery1.sql*'. The query is:

```
SELECT ProductName, QuantityPerUnit, UnitPrice, ReorderLevel
FROM Products
```

The results grid displays the following data:

	ProductName	QuantityPerUnit	UnitPrice	ReorderLevel
1	Chai	10 boxes x 20 bags	18.00	10
2	Chang	24 - 12 oz bottles	19.00	25
3	Aniseed Syrup	12 - 550 ml bottles	10.00	25
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22.00	0

- Click on the **Column Mappings** tab. In doing so, you open the **Column Mappings** page in the **Advanced Editor for DataReader Source**, showing the columns that are the output of the DataReader.

For this tutorial, no modifications are made to the **Input and Output properties** page.

Step 4: Adding an OLE DB Destination and Establishing a Path from the DataReader Component

In this step, we will add an *OLE DB Destination*. We will then establish a path from the *DataReader Source* to the *OLE DB Destination*.

- Add an **OLE DB Destination** component from the **Data Flow Destinations** group in the toolbox to the **Data Flow** page.

This can be accomplished either by a double-click of the component in the **Toolbox** or a drag-and-drop operation as described in previous chapters.

- Right-click **DataReader Source**. From the drop-down menu choose, **Add Path**.

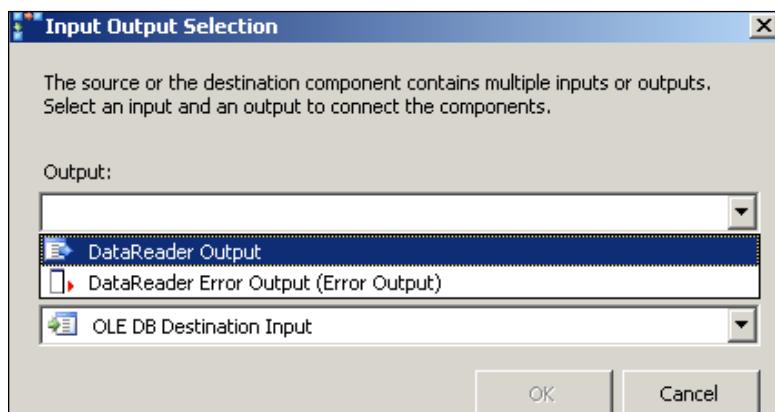
This opens up the window, **Data Flow**, which allows you to establish a data flow path, and displays the "From:" location as **DataReader Source**.

3. Click on the drop-down along "To:".

This will show both the OLE DB Destination as well as the DataReader Source.

4. Choose the **OLE DB Destination** and click on the **OK** button..

This opens up the **Input Output Selection** window that shows **output** and **input** windows available. You need to choose the appropriate ones. The path should connect from the **DataReader Source** to the **OLE DB Destination Input** (this will be the default), as shown in the following screenshot.



5. Choose the above options, click on the **OK** button on this screen.

You will see a green line connecting the **DataReader Source** to the **OLE DB Destination**.

Step 5: Configuring the OLE DB Destination Component

The OLE DB Destination also requires a connection manager. The **OLE DB Destination** connects to an MS Access Database file on your hard drive using the connection properties defined in a connection manager.

1. Right-click **OLE DB Destination** and from the drop-down menu click on **Edit....**

You get a **Microsoft Visual Studio (Standard Edition 8.0.50727)** warning ("This Component has no available...").

2. Click on the button **Yes**, to this warning.

This opens the **OLE DB Destination Editor** with a list on the left-hand side and three OLE DB related drop-down boxes on the right.

3. Click on the **New...** button right next to the first drop-down.

This will look for a New OLE DB connection manager by opening the **Configure OLE DB Connection Manager** window. This window may display existing OLE DB connections in the left-hand side under the label "**Data Connections:**".

4. Click on the **New...** button.

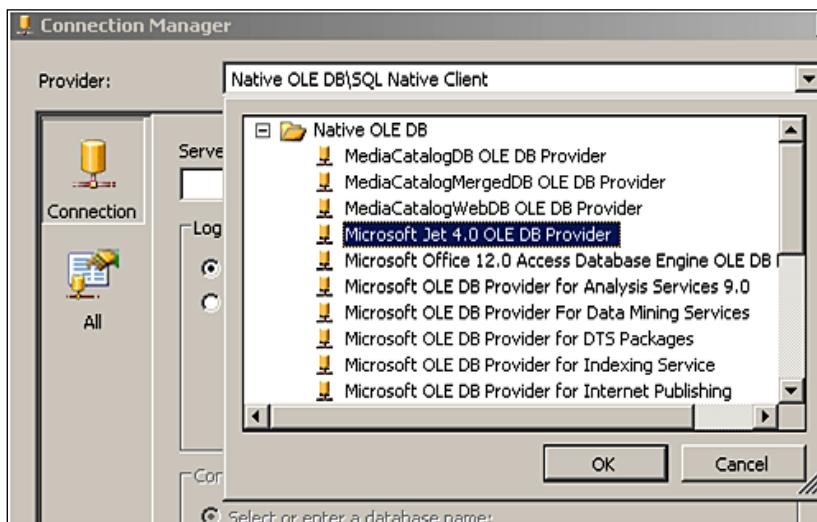
This opens the **Connection Manager** window shown in the following screenshot.



Microsoft Access Databases are accessed by Microsoft Jet 4.0 OLE DB providers.

5. Click on the first drop-down along the label, "**Providers:**".

This will display a list of *providers*.



6. In this list, click on the Microsoft Jet 4.0 OLE DB Provider.

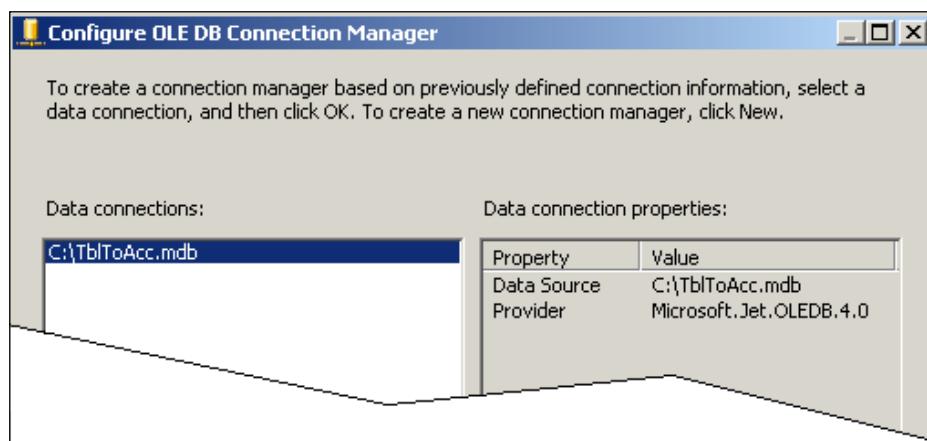
Now you need to provide a database file name. You may browse and choose an existing MS Access database file, or create one inside the window of the Open dialogue and choose that file.

7. For this exercise, create a file TblToAcc .mdb and choose this file while browsing.

This action will get the path to this file into the **Connection Manager** window.

Admin is the default username and it does not need a password as the administrator is also the present user. You may also test the connection with the **Test Connection** button. You should get a connection succeeded message from the connection manager.

8. Click on the **OK** button in the **Connection Manager** window. This adds the Data Connection to the **Configure OLE DB Connection Manager** window, as shown in the following screenshot.

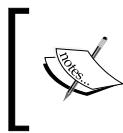


9. Click on the **OK** button in the **Configure OLE DB Connection Manager** window.

This adds the **OLE DB Connection Manager** with the same name as that of the Mdb database file to the OLE DB Destination Editor, **TblToAcc**, as shown in the following screenshot.

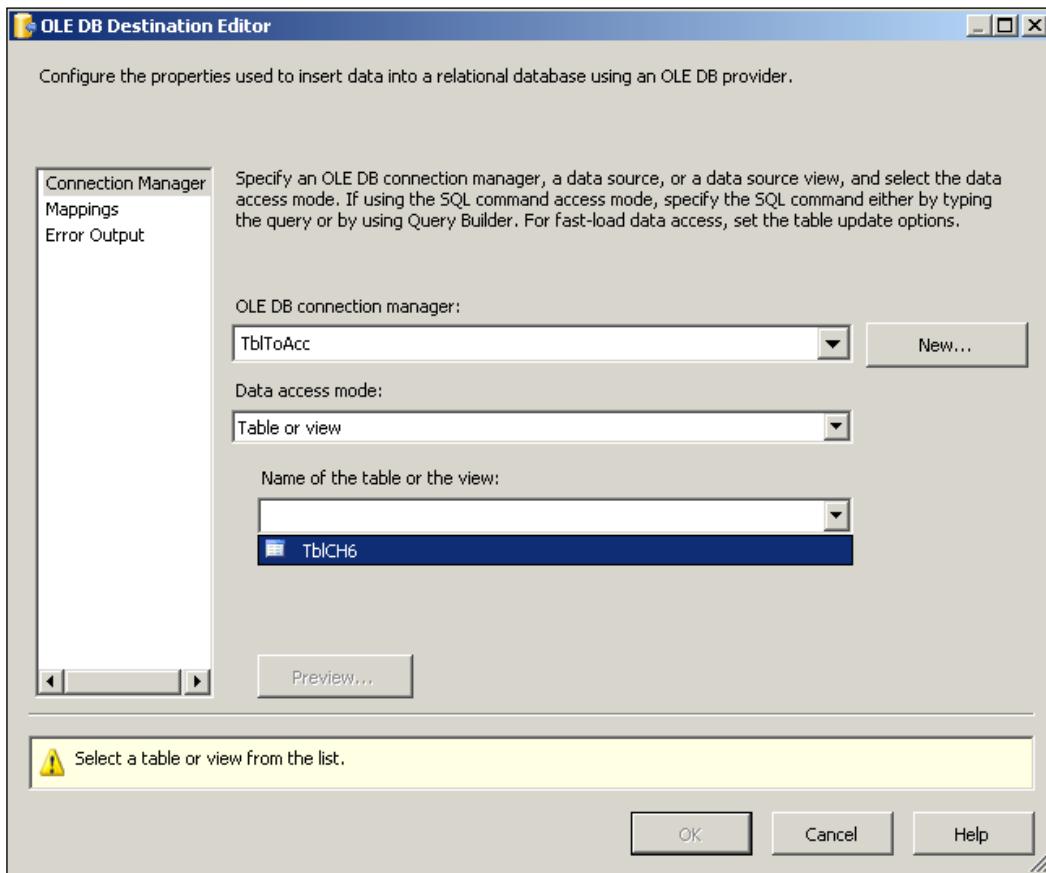
10. In the third drop-down in this editor, (**OLE DB Destination Editor**), click on the drop-down arrow along the label, **Name of the Table or View:**.

It will try to load but there are no tables or views as the table is empty except for system tables.



Depending on the version of VS 2005, the OLE DB Destination Editor may display a **New...** button alongside **Name of the table or the view** label. This will allow you to create a new table instead of creating it manually, as described.

11. Open the MDB file and keep only four columns (in Table View) and save the table as **TblCH6** after renaming the columns as Product Name, Quantity/Unit, Price/Unit, and Level.
12. Now click on the drop-down one more time. This time you will see the table **TblCH6**, as shown in the following screenshot.

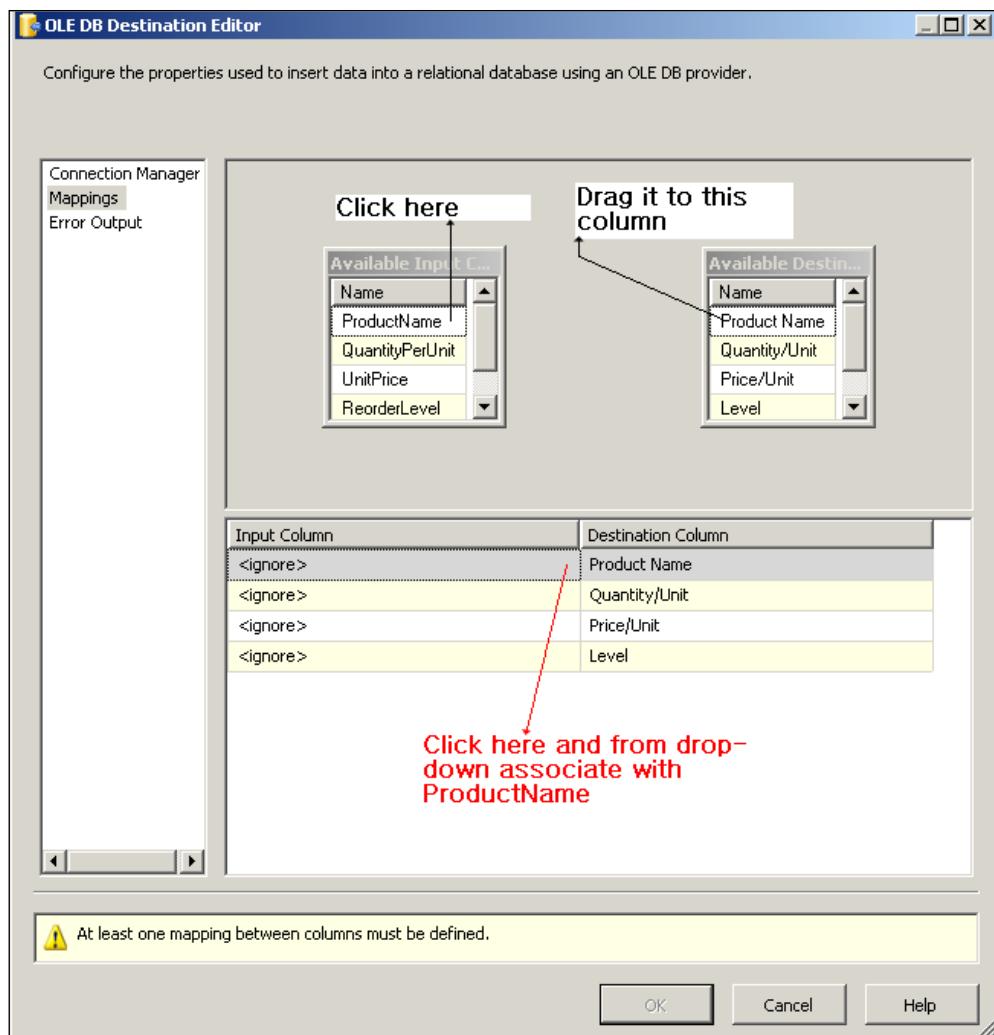


13. Choose this table in the drop-down.

14. Next, click on the **Mappings** list on the left-hand side.

This opens the **Mappings** window. Either you can associate the input and output columns by clicking on the input and dragging it to the corresponding output in the design as indicated, or click on the indicated area in each of the four rows in the tabulated area below the screen, and associate inputs and outputs as shown.

15. Associate **Available Input Columns** with **Available Destination Columns** as suggested schematically in the figure (as explained above). All four columns need to be associated in this manner.



At present, we are not considering any error outputs.

16. Click on the **OK** button in this screen.

This completes the extraction from the SQL Server, and loading it to the table in the Access database. This package can now be executed.

17. Build and execute this package as in the previous exercises.

When executed, you will notice that both the data flow components turn green and you can verify that the **TblToAcc** is now populated with data, as shown partially in the next screenshot.

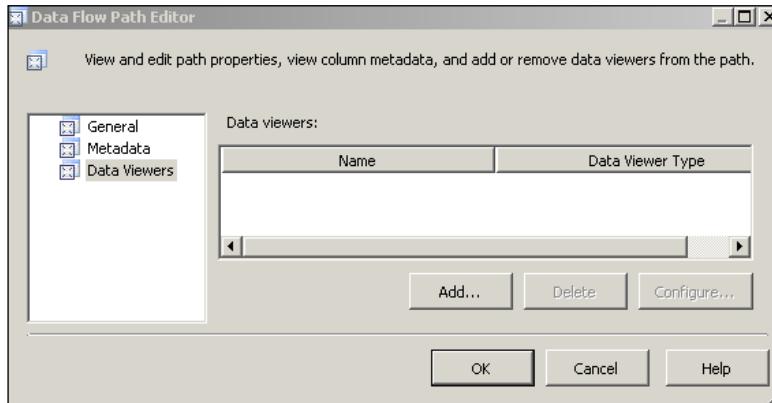
TblCH6 : Table				
	Product Name	Quantity/Unit	Price/Unit	Level
►	Chai	10 boxes x 20 bags	18.000000000000000000	10
	Chang	24 - 12 oz bottles	19.000000000000000000	25
	Aniseed Syrup	12 - 550 ml bottles	10.000000000000000000	25
	Chef Anton's Ca...	48 - 6 oz jars	22.000000000000000000	0
	Chef Anton's Gu...	36 boxes	21.350000000000000000	0
	Grandma's Boysen...	12 - 8 oz jars	25.000000000000000000	25
	Uncle Bob's Organic...	12 - 1 lb pkgs.	30.000000000000000000	10
	Northwoods Cranberry...	12 - 12 oz jars	40.000000000000000000	0
	Mishi Kobe Niku	18 - 500 g pkgs.	97.000000000000000000	0
	Ikura	12 - 200 ml jars	31.000000000000000000	0
	Queso Cabrales	1 kg pkg.	21.000000000000000000	30

Step 6: Incorporating a Data Viewer to Monitor Data Flow

Data Viewers in SSIS display data flowing in a data path connecting two data flow components by stopping the flow at the point in which they are placed. The data flow can be resumed by controls available on the Data Viewer. The data can be viewed in four different ways, in a grid, a histogram, a scatter plot, and a column chart. Data Viewers are excellent trouble shooting devices as they enable monitoring the flow and diagnosing the bottle necks.

1. Right-click the green line, the Path connecting the **DataReader Source** and the **OLE DB Destination** and from the drop-down menu choose **Data Viewers....**

This brings up the following window, **Data Flow Path Editor**.



2. Click the **General** list item on the left in the above window.

Verify that it shows the Common and Miscellaneous properties of the path.

3. Click on the **Metadata** list item on the left in the above window.

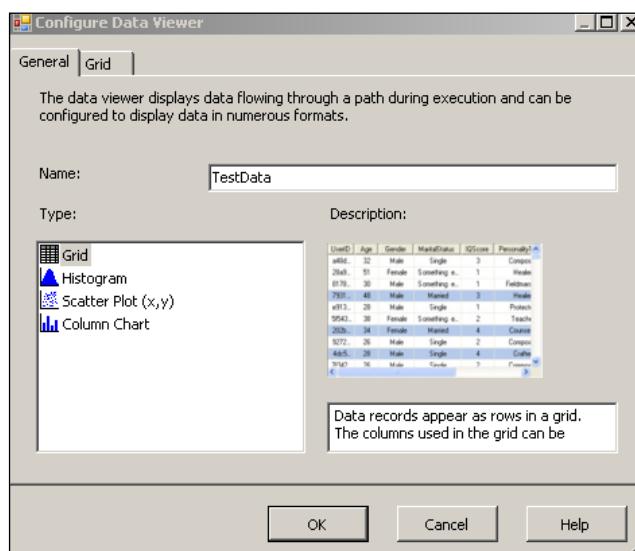
Review the Metadata page on the right, showing the column Metadata from the DataReader Source.

4. Click on the **Data Viewers** item.

As there are no Data Viewers added, the page is empty.

5. Click on the **Add...** button in the above window to add a Data Viewer.

The **Configure Data Viewer** window is displayed.



The **Grid** type is the default.

- Type in the name **TestData** for the "Name:" field. Then click on the **OK** button.

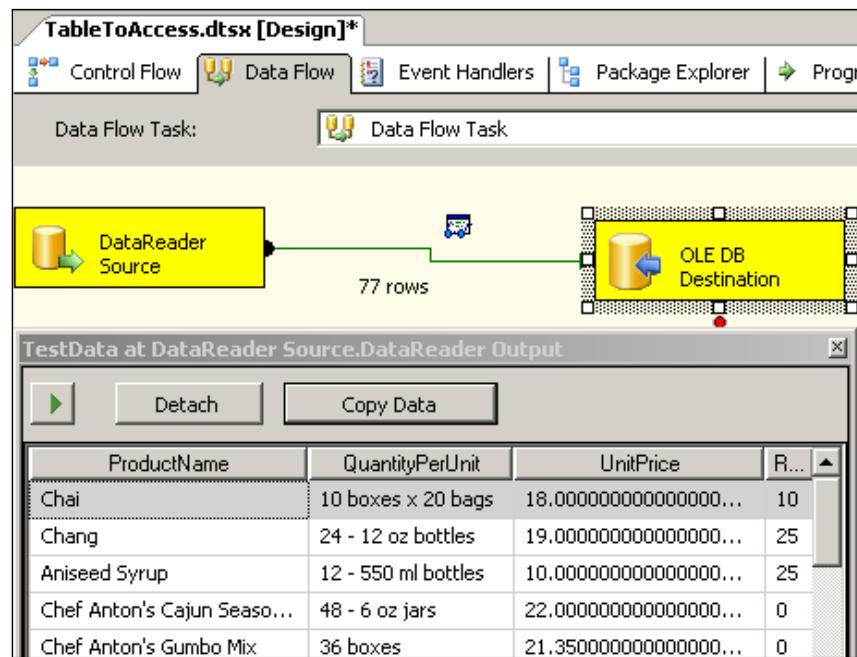
Now you will see a Data Viewer *TestData* in the "Data Viewers:" window.

- Click on the **OK** button on this screen.

This adds a little icon, the Data Viewer icon as shown, in the proximity of the green path line in the Canvas.

- Now build the project and execute the package.

The package runs and the **Data Viewer** displays the data that would flow to the destination, as shown in the next screenshot. The Data Viewer is like a break (or stop) in the flow of data, the package is not fully executed at this stage, and the source and destination icons are yellow.



- Click the green arrow head in the **TestData at DataReader Source. DataReader Output** screen.

By clicking the green arrow head, or by toggling the **Detach** button, the data flow can be resumed past the Data Viewer. The data gets to the destination, and the source and destination icons turn green. In the above screenshot, the Data Viewer is showing the data at the output terminals of the DataReader.

Summary

This chapter described extracting data from an SQL Server 2005, and loading it to the MS Access Database using the Microsoft Jet 4.0 OLE DB Provider. This chapter also showed how to use a Data Viewer to monitor the flow in the data path.

7

Data Transfer from a Text File Using the Bulk Insert Task

This chapter shows you how to create a package that can transfer data from a text file to a table in a SQL Server database using the Bulk Insert Task.

The Bulk Insert Task is used when you need to transfer a large amount of data residing in flat files onto a table in SQL Server 2005's database. In previous versions of SQL Servers, this was the method of choice to transfer large amounts data rapidly, utilizing an executable program called `BCP.exe` (Bulk Copy). This utility is also available in SQL Server 2005 and works in either direction, from a SQL Server table (view) to a data file in a user-specified format, or from a data file to a SQL Server table (view). However, the Bulk Insert Task can only transfer from a flat file to a table (view) in a SQL Server.

The Bulk Insert Task cannot be used with any of the data flow transformation components, and in this respect it is not as versatile as a data flow task. This task has been provided for backward compatibility.

Hands-On Exercise: Transferring Data from a Flat File to a SQL Server Database Table

In order to follow the steps as indicated, you will need access to a table in a database on the SQL Server 2005 as well as knowledge of the nature (format) of the file and its location.

The following are the major steps in this exercise:

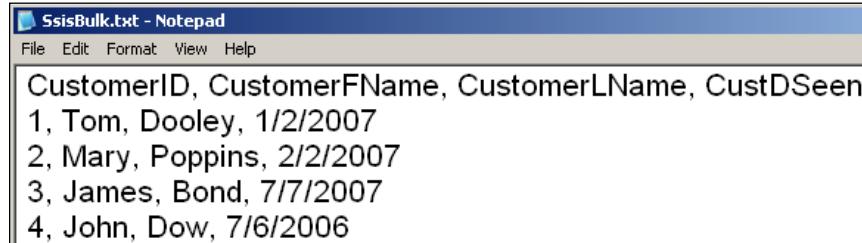
- Use/create a flat text file whose contents need to be transferred.
- Create a SQL Server database table with columns that can accept the contents of the file created in the previous step.
- Create a BI Project and add a Bulk Insert Task.
- Configure the Bulk Insert Task.
- Build and execute the package.

Step 1: Use / create a Flat Text File whose Contents Need to be Transferred

Normally, this file should be available, as this is the starting point. Since this is just a demo, we will be using a file with 10 rows, of which the 1st row is a column header, a ludicrously small file for this heavy weight tool. This file can be created with a text editor such as *Notepad*, but usually, it is resident in legacy data stores.

1. Create a text file in Notepad called `SsisBulk.txt` and save it on `C:\` drive.

The following screenshot shows the contents of this file. You may create your own text file (alternatively, a suitable CSV file such as the one from an earlier chapter can also be used). The following screenshot shows a couple of rows of this file. In practice, however, large amounts of data with tens of thousands of lines of data can be transferred.



Step 2: Create a Table with Columns that Can Accept the Contents of the File Created

The data in the above text file will be transferred to a table in the *MyNorthwind* database. This table will have four columns with the columns headings: CustomerID, CFname, CLname, and CRdate. This can be created by the statement shown in the following screenshot, in SQL Server 2005.

```
(local).MyNort...SQLQuery1.sql* Summary
Create Table[dbo].[TestCustomer] (
    [CustomerID][int] Identity(1,1) not null,
    [CFname][nchar](20) null,
    [CLname][nchar](25) null,
    [CRdate][smalldatetime]null,
    Constraint[PK_TestCustomer] Primary Key Clustered
    ([CustomerID]Asc) ) on [Primary]
```

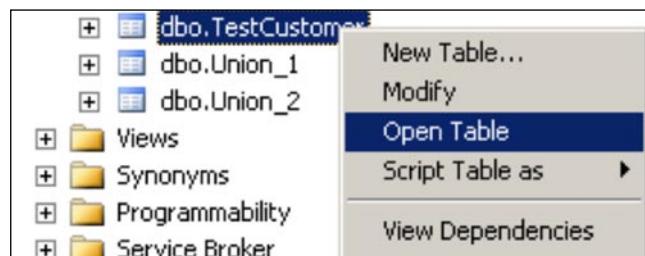
Verify SQL Syntax

Messages

Command(s) completed successfully.

```
Create Table[dbo].[TestCustomer] (
    [CustomerID][int] Identity(1,1) not null,
    [CFname][nchar](20) null,
    [CLname][nchar](20) null,
    [CRdate][smalldatetime]null,
    Constraint[PK_TestCustomer] Primary Key Clustered
    ([CustomerID]ASC) ) on [Primary]
```

1. Open the **Query** window in the Microsoft SQL Server Management Studio and type in the above statement.
2. After verifying the SQL Syntax by clicking on the verify syntax symbol, click on the **Execute** symbol (!).
3. Now go to the *MyNorthwind* database, in which this table is created. Make a right click on the Tables node, and from the drop-down choose Refresh.
4. Expand the Tables node. Locate the table, TestCustomer. Make a right click on the TestCustomer table and choose Open Table.



This displays the TestCustomer table in the management studio as shown in the following screenshot.

Table - dbo.TestCustomer					(local).MyNort...SQLQuery1.sql*	Summary
	CustomerID	Cfname	CLname	CRdate		
*	NULL	NULL	NULL	NULL		

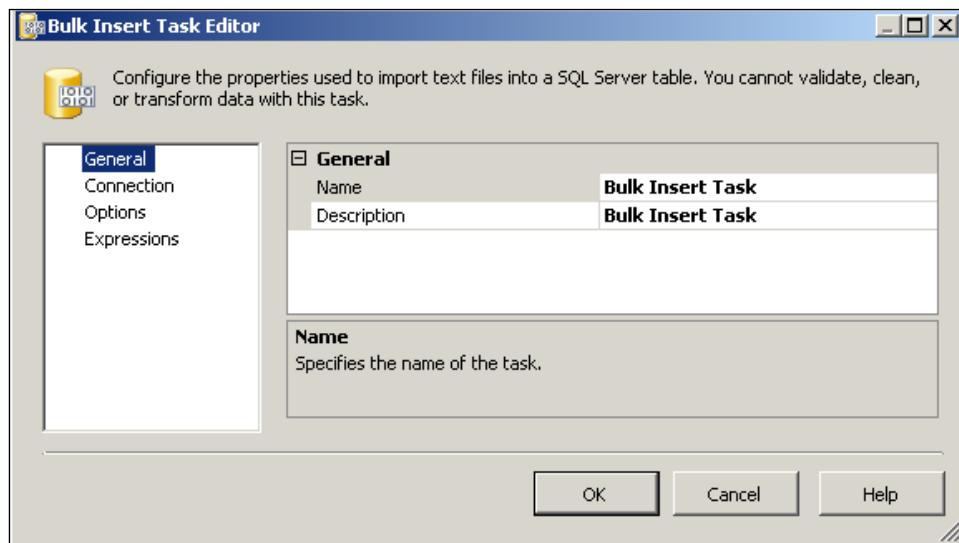
Step 3: Create a BI Project and Add a Bulk Insert Task

1. Create a Business Intelligence project **Ch7** and change the default name of the package from Default.dtsx to BulkInsert.dtsx.
2. Drag and drop a **Bulk Insert Task** from the **Control Flow** items group, or by double-clicking the **Bulk Insert Task** in the **Toolbox** when the **Control Flow** page of the canvas is highlighted.

This brings in the **Bulk Insert Task** into the canvas. This icon is displayed with a white cross in a red circle indicating that its configuration is not complete.

Step 4: Configure the Bulk Insert Task

1. Right click the Bulk Insert Task and from the drop-down choose **Edit....**
This brings up the **Bulk Insert Task** editor as shown in the following screenshot.

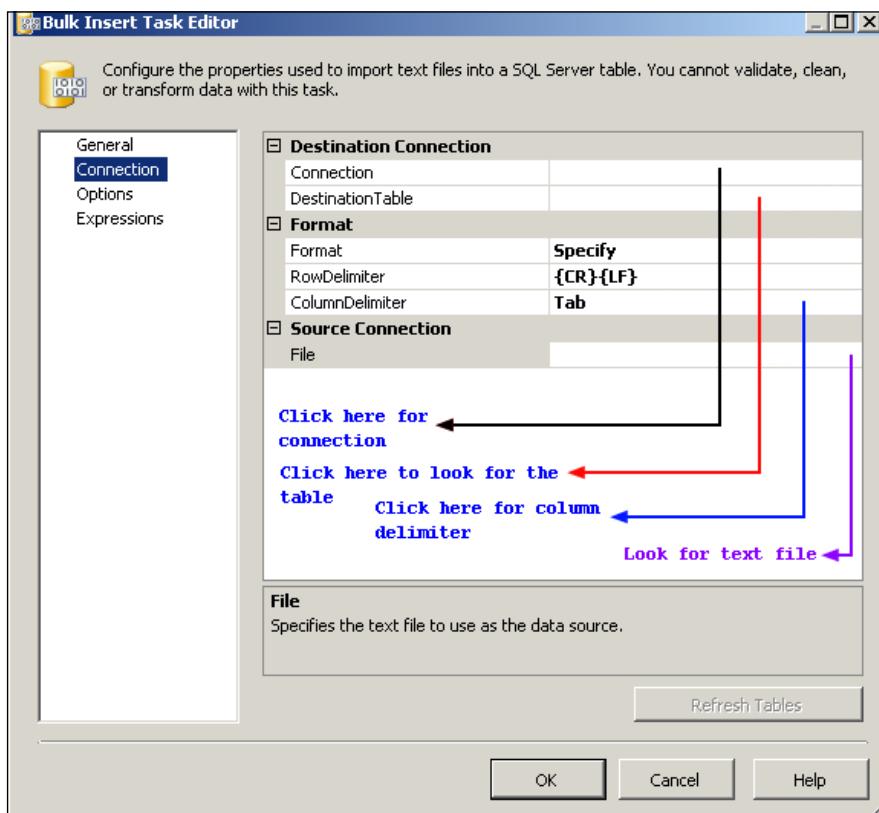


2. Provide a meaningful **Name** and **Description** by changing the default text values shown in the previous screenshot.

For the **Name** field, "Bulk Insert TestCustomer Task" is chosen. For the **Description** field, "Insert Customer Info from a text file" is chosen for this exercise.

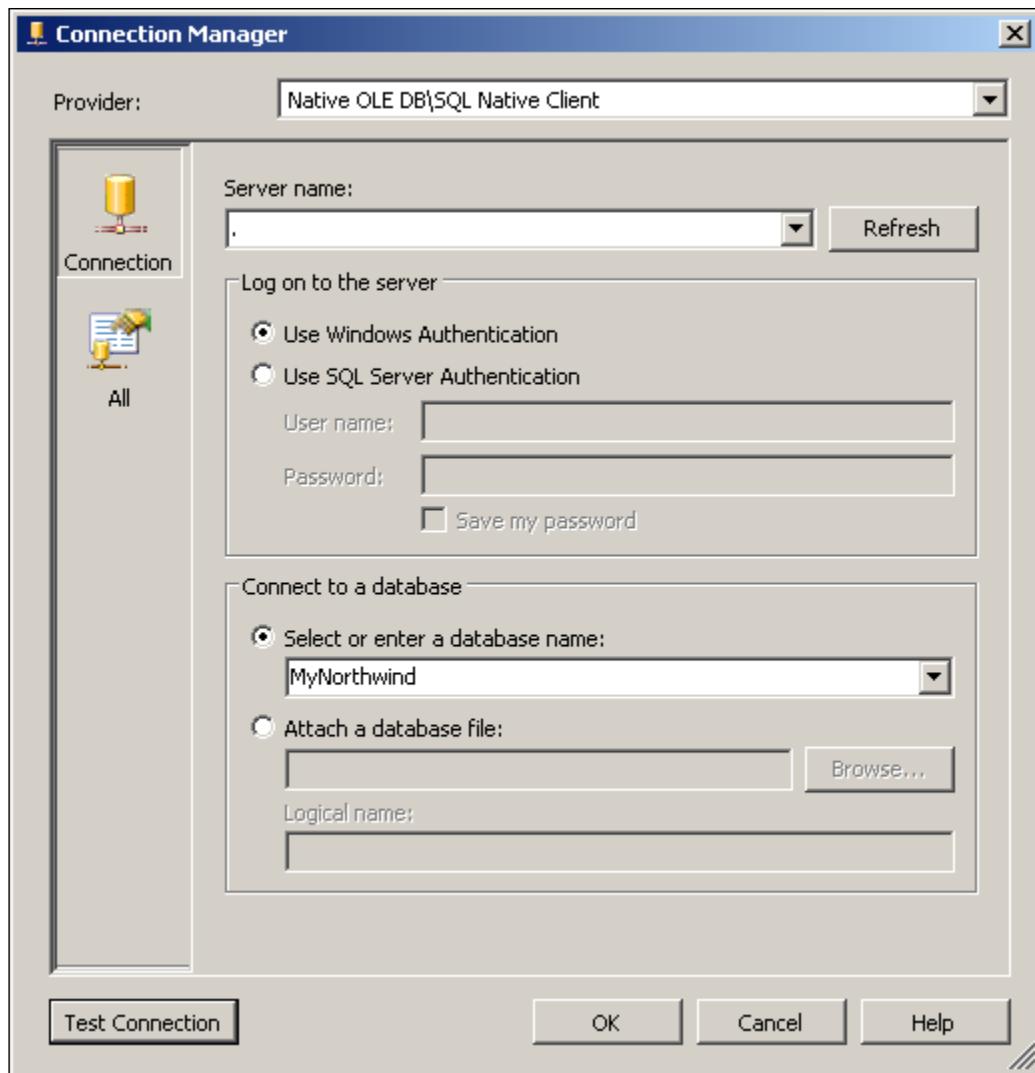
3. Click on the **Connection** list item in the left-hand-side.

This opens the Connection page as shown in the following screenshot. This has three major nodes: **Destination Connection**, **Format**, and **Source Connection**. The destination is the SQL Server. Here, you should specify the table created earlier. The **Format** node has three items: **Format**, **RowDelimiter**, and **ColumnDelimiter**. Format has two options, **Specify** and **Use File**. The **Specify** option will be used. The **RowDelimiter** and **ColumnDelimiter** specify how the rows and columns are delimited in the source text file. For this example, only the **ColumnDelimiter** was changed from **tab** (default) to **Comma{,}** to match the text file. In the **Source Connection** node, you will be required to locate the text file you created earlier.

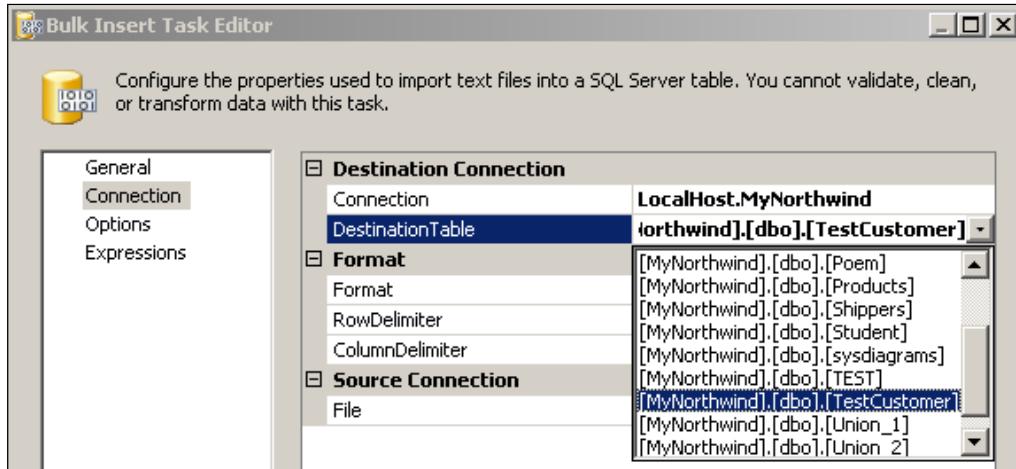


4. Click on the area shown for making a connection. Click on the drop-down handle, and from the menu choose **New Connection....**
5. In the OLE DB Connection Manager that shows up, click on the **New...** button.

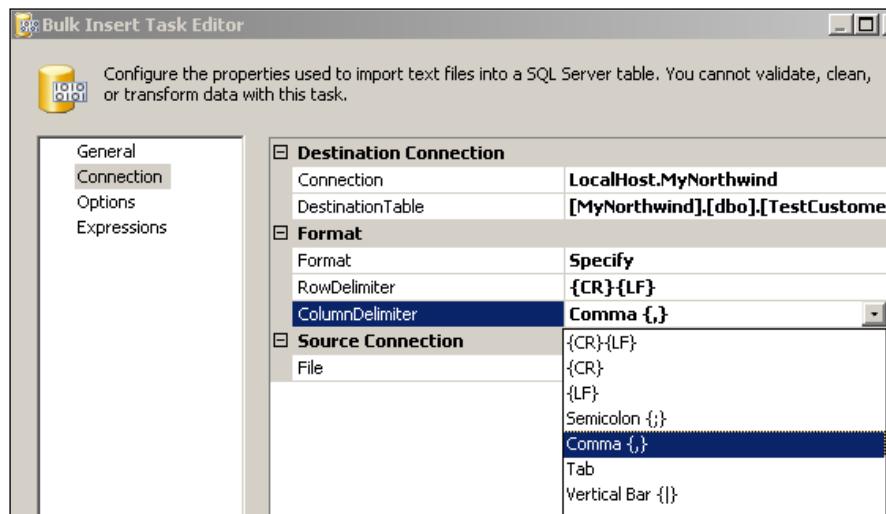
This opens the **Connection Manager** window as shown in the following screenshot. This was the same one we saw in the earlier chapters (Chapter 4, 5). The same choices are used. The connectivity may be tested with the **Test Connection** button. Notice that the **Provider** is the new type, **Native OLE DB\SQL Native Client**.



6. Fill in all the details as shown in the previous screenshot. Verify the connectivity is good by clicking the **Test Connection** button.
7. Click on the **OK** button (shown in the previous screenshot).
This brings in **localhost.MyNorthwind** into the **Connection** textbox.
8. Now click on the next area for choosing the table.
From the drop-down that shows a list of tables choose **TestCustomer** table, as shown in the following screenshot.

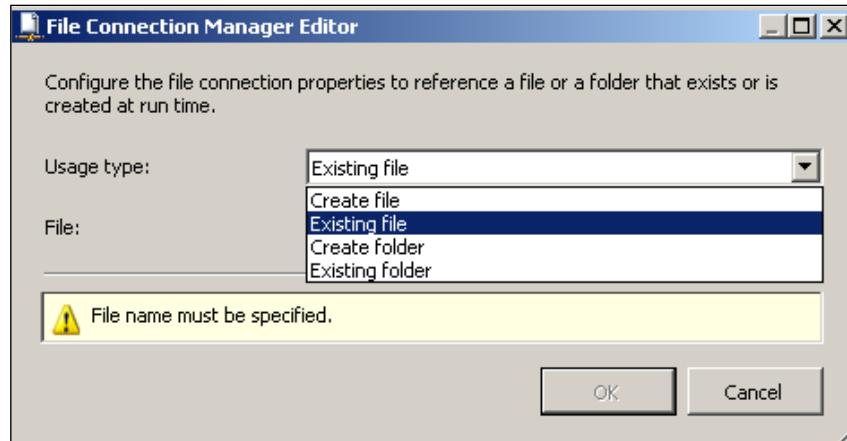


9. Click on the area near **ColumnDelimiter** in the **Format** node, and change from **Tab** to **{,}**, as shown in the following screenshot.



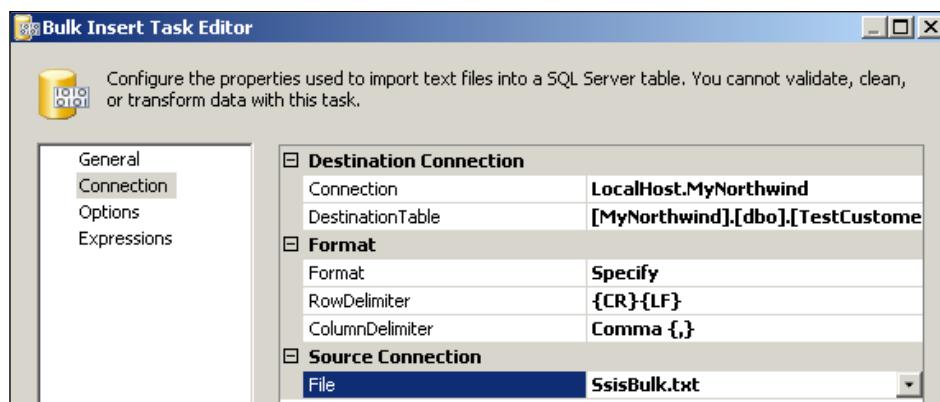
10. Click on the area shown near **File** to look for the file created in **Step 1**.
This reveals a drop-down.
11. Click on the drop-down arrow head and choose **New Connection....** Accept the default **Existing File**.

This opens the **File Connection Manager Editor** as shown in the following screenshot. It has four options that can be seen by clicking on the drop-down.



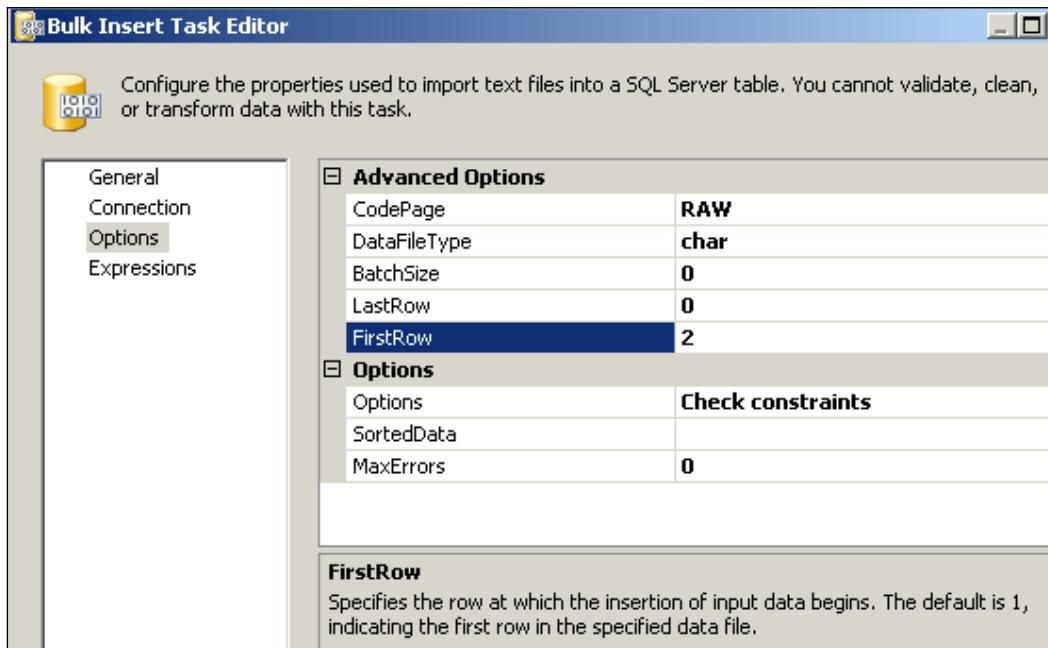
12. Use the default, **Existing File**. Click on the **Browse...** button, which opens a **Select File** dialog. Locate the file created earlier in the C:\ drive, SsisBulk.txt and click on the Open button in that dialog.
13. Click on the **OK** button in the **File Connection Manager Editor** that shows up again.

This will add the file information to the **Bulk Insert Task Editor** window, as shown in the following screenshot.



14. Click on the list item **Options**.

This opens the following window where there are several options you can choose as shown. Here, only the **FirstRow** will be changed from 1 to 2. This is because insertion into table begins from the second row in the text file, as the first row contains column headers.



15. Click on the **OK** button on the **Bulk Insert Task Editor**'s window.

This completes the configuration of the Bulk Insert Task. In this exercise, the **Expressions** branch is not configured because the focus is on a simple Bulk Insert Task configuration.

Step 5: Build and Execute the Package

1. Build the project and execute the package.

The **Bulk Insert Task** object first turns yellow and then to green, indicating that the task has successfully executed.

2. From the **Debug** menu item, choose **Stop Debugging**.

3. In the SQL Server Management Studio, review the table dbo.TestCustomer.

You will see the contents of the table as shown in the following screenshot.

You may also review the tabbed page, **Progress**, on the Canvas to look at the details of the processing steps.

Table - dbo.TestCustomer				
	CustomerID	CFname	CLname	CRdate
▶	1	Tom	Dooley	1/2/2007 12:00:...
	2	Mary	Poppins	2/2/2007 12:00:...
	3	James	Bond	7/7/2007 12:00:...
	4	John	Dow	7/6/2006 12:00:...
	5	Bobby	Darin	1/1/2005 12:00:...
	6	Charles	Azanour	... 5/5/2005 12:00:...
	7	Aishwarya	Roy	2/3/2003 12:00:...
	8	Ned	Kelly	4/5/2007 12:00:...
	9	Diego	Maradona	... 9/1/2001 12:00:...
*	NULL	NULL	NULL	NULL

What Happens if there Is an Error?

Format errors can exist in the text file. For example, the second row may have an error in the date format (2//2/2007 instead of 2/2/2007). When this file is used for inserting, the Bulk Insert Task turns red. A review of the tabbed page **Progress** indicates that there was a data conversion error flagged, as follows:

"[Bulk Insert Task] Error: An error occurred with the following error message: "Bulk load data conversion error (type mismatch or invalid character for the specified codepage) for row 3, column 4 (CRdate)".

However, you should notice that except for that row ("Mary Poppins" is missing), all other rows are inserted into the table as seen in the Microsoft SQL Server Management Studio shown in the following screenshot. Make sure you refresh the server objects to see the changes as well as stop debugging the project in Visual Studio 2005.

Table - dbo.TestCustomer				
	CustomerID	Cfname	Clname	CRdate
▶	10	Tom	Dooley	1/2/2007 12:00:...
	11	James	Bond	7/7/2007 12:00:...
	12	John	Dow	7/6/2006 12:00:...
	13	Bobby	Darin	1/1/2005 12:00:...
	14	Charles	Azanour	... 5/5/2005 12:00:...
	15	Aishwarya	Roy	2/3/2003 12:00:...
	16	Ned	Kelly	4/5/2007 12:00:...
	17	Diego	Maradona	... 9/1/2001 12:00:...
*	NULL	NULL	NULL	NULL

Summary

The Bulk Insert Task configuration was described in detail. Input text files can have various types of errors. The error messages can be used to debug the errors. Bulk Insert Task used the default format option "Specify" for loading the text file. It has another option, "use file", which would require a format file. The format file can be generated using the SQL Server 2005's BCP.exe utility. If any data transformation is needed, then a better option is to use the Data Flow Task.

8

Using a Conditional Split Data Transformation

This chapter shows you how to create a package that can split data based on a given set of criteria into separate flows, a conditional split of data. You will also learn how to use the in-memory data flow destination, the Recordset Destination.

Often, you want to separate out data meeting certain established criteria and direct them to different destinations. The data splitting may be needed for any number of reasons, to focus on a business objective: weed out data that are suspicious; data that does not follow some standard pattern; identify missing data, etc. The data flow transformation, Conditional Split Transformation is ideally suited for such an operation.

Hands-On Exercise: Splitting Data Retrieved from a SQL Server

In order to follow the steps as indicated, you will need a data flow task that connects to a data source, and destinations to which the separated flows can be directed into, which includes a destination for the default flow that does not meet the criteria established for data splitting. You will also need to add and configure an in-memory ADO recordset destination, the Recordset Destination component to visualize the data.

The following are the major steps of this exercise:

- Create a BI Project and add a Data Flow Task. Add and configure the DataReader Source to pull data from the Local SQL Server.
- Add a Conditional Split Transformation.

- Establish a path to Connect DataReader Source with the Conditional Split Data Transformation.
- Configure the Conditional Split Data Transformation.
- Add Recordset Destination(s).
- Configure the Recordset Destination(s).
- Build, execute the package, and review results.

Step 1: Create a BI Project and Add a Data Flow Task. Add and Configure the DataReader Source to Pull Data from the Local SQL Server

Creating a BI project, changing the name of the default Package, adding a DataReader Source, and configuring it to provide at its output data selected from the database has been described in the previous chapters.

1. Create a BI project **Ch 8** and change the default name of the package to `splitter.dtsx`. Add a DataReader Source and provide it with a connection manager [`localhost.MyNorthwind.sa`] that retrieves data from the Local SQL Server.
2. Configure the DataReader Source with the following SQL statement for retrieving the data. **Select * from [Order Details]**.
3. The following screenshot shows the result of running this query directly in the query window of the SQL Server Management Studio.

The screenshot shows the SQL Server Management Studio interface. The title bar says '(local).MyNort...SQLQuery1.sql*'. The query window contains the SQL command: 'Select * from [Order Details]'. Below the query window, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays a table with the following data:

	OrderID	ProductID	UnitPrice	Quantity	Discount
1	10248	11	14.00	12	0
2	10248	42	9.80	10	0
3	10248	72	34.80	5	0
4	10249	14	18.60	9	0
5	10249	51	42.40	40	0
6	10250	41	7.70	10	0
7	10250	51	42.40	35	0.15

Step 2: Add a Conditional Split Transformation

The data output from the DataReader Source will be the input to the conditional split transformation. In a **Conditional Split Data Flow Transformation**, the data flow is read and depending on the conditions and their order of implementation you define, the data is split into components that satisfy those conditions. Those that do not satisfy are separated out into another data flow. It acts very much like a sieve sizing the grains, or in basic computer terms it is based on the **case** statement. This was explained in Chapter 1.

In this present example, our condition for splitting will be based on the value in the **Discount** column shown in the previous screenshot. Specifically, we separate out the data in the table based on those that have a discount > 0 and those that have discount ≤ 0 . These are the Boolean conditions that will be implemented.

1. Drag and drop a **Conditional Split** Data Flow Transformation from the **Data Flow Transformations** group in the **Toolbox** to the **Data Flow** page of the canvas, 'Canvas'.

Step 3: Establish a Path to Connect DataReader Source with the Conditional Split Data Transformation

1. Right click the **Data Reader Source** and from the drop-down click on **Add Path**.

This displays the **Data Flow** window with the "From:" showing **Data Source Reader**. The process of establishing a path is same as in the previous chapters.

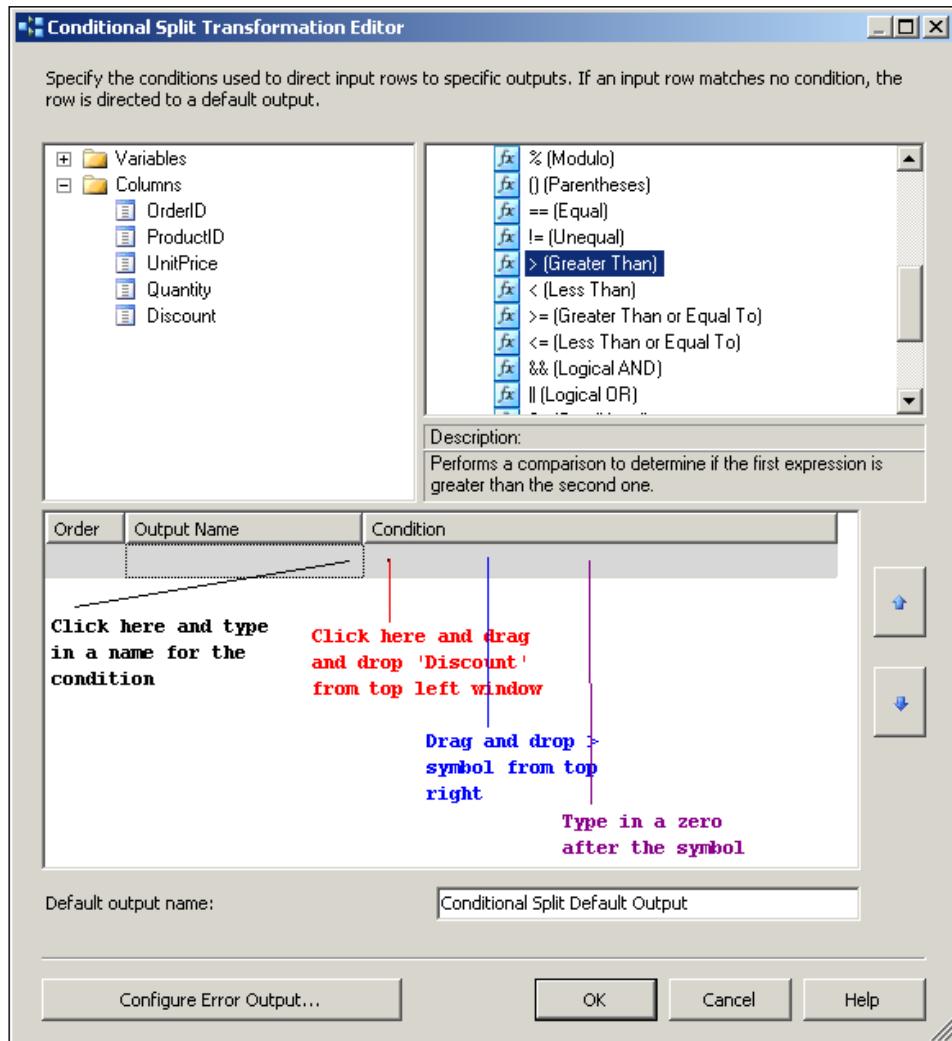
When establishing the path is completed, you should be able to see a thin green line from the **Data Reader Source** to the **Conditional Split** transformation in the canvas.

Step 4: Configure the Conditional Split Data Transformation

1. Right click the **Conditional Split** component on the canvas and from the drop-down menu choose **Edit....**

This opens the **Conditional Split Transformation Editor**. Read the instructions in this window. It has three main areas with **Variables** and **Columns** in the left window and some standard core items in the right window. The bottom portion of the window is where the data splitting conditions are developed by drag and drop operations. The variables used

in splitting, and the operation needed to fashion the split can both be dragged and dropped. The way the splitting is configured is shown in annotation. In this part of the window, the name for the split is provided in a drop-down list. There is also a provision to order the split by using the arrows on the right, and a button that will take you to configure the default of a conditional split transformation.



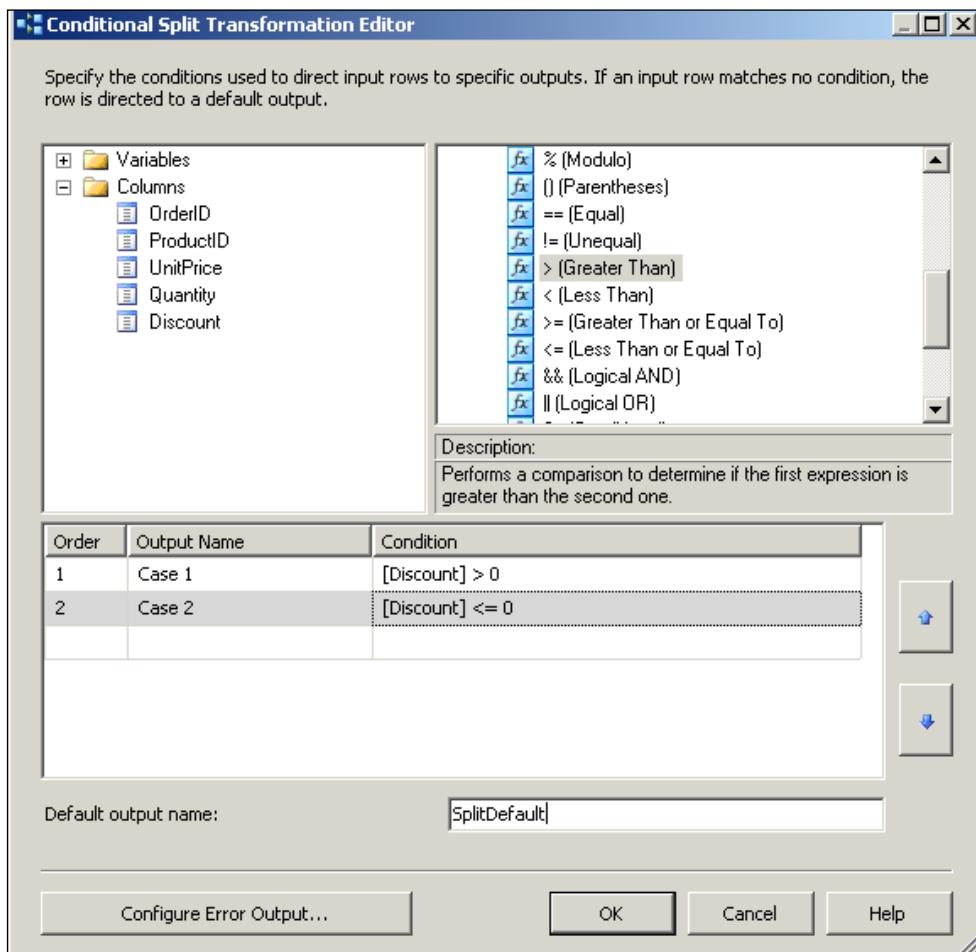
2. Click under **Output Name** and type in the name "Case 1" for this example, the **Order** field gets populated with 1.

3. Now following the schematic (red, blue, magenta instructions) in the above screenshot, format **Discount > 0** in the **Condition** column.
4. Using the same procedure, create Case 2 below Case 1 with **Discount <= 0**.

By default, Case 1 will be the first and Case 2 is the second in order. This can be changed by using the up/down arrows at the far right of the window.

5. Provide the name, "SplitDefault", for the **Default Output** name.

In this particular example, we will not configure an error output. The completed window appears as shown in the following screenshot.



6. Click on the **OK** button to complete the Conditional Split Transformation. The Conditional Split Transformation will be producing three outputs, one each for **Discount > 0**, **Discount <= 0**, and the default that is neither of the two.

Step 5: Add Recordset Destination(s)

Although the three streams of data can be channeled to any of the different **Data Flow Destination** types that we have already used in the previous chapters, we will do something a little different here.

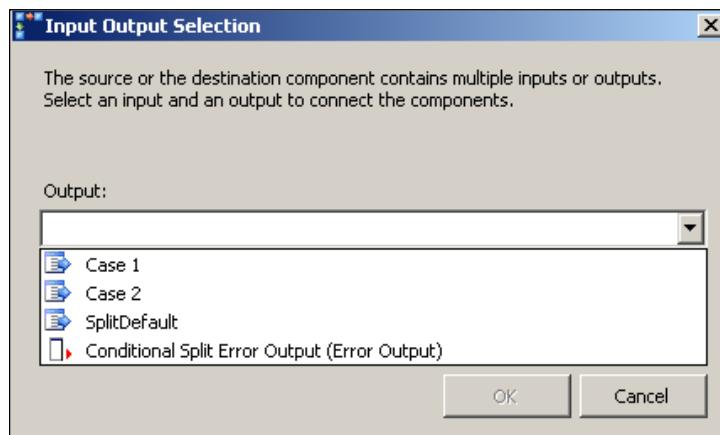
When a Recordset Destination component is used in the SSIS designer, an in-memory ADO (ActiveX Data Objects) recordset is created and populated. A **Recordset Destination** requires a variable to be defined where the ADO recordset is going to be stored. The variable name is a string you specify at design time. You may use this variable outside the data flow to be consumed by other elements or used in scripts. It has one input and no error output.

1. Drag and drop (or alternately double-click in the **Toolbox**) a **Recordset Destination** component into the **Data Flow** page of the canvas. "Canvas".
2. Right-click the **Conditional Split** component and choose **Add Path**.

This opens the Data Flow window displaying the "From:" as Conditional Split.

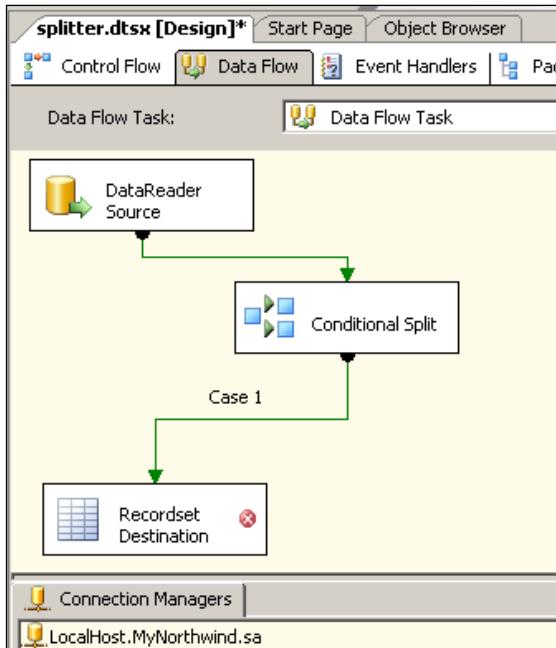
3. Click on the drop-down arrow head in the "To:" window. In the drop-down, you will see three options. Choose **Recordset Destination**. Click on the **OK** button in the **Data Flow** window.

This opens the **Input Output Selection** window with the input displaying "Recordset Destination" and the output from the conditional shows all the previously configured splits as shown in the following screenshot.



4. Click on **Case 1** and click on the **OK** button in the **Input Output Selection** window.

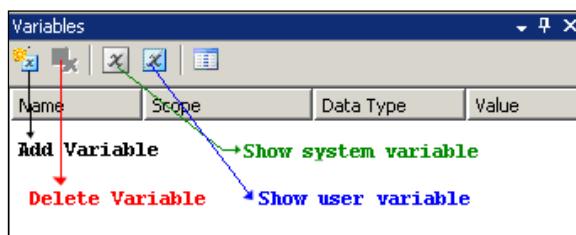
5. At this point, the canvas appears as shown in the following screenshot. The Recordset Destination is not completely configured as yet.



6. Similar to the above, add two more Recordset Destinations, one each for "Case2" and the "SplitDefault" data.

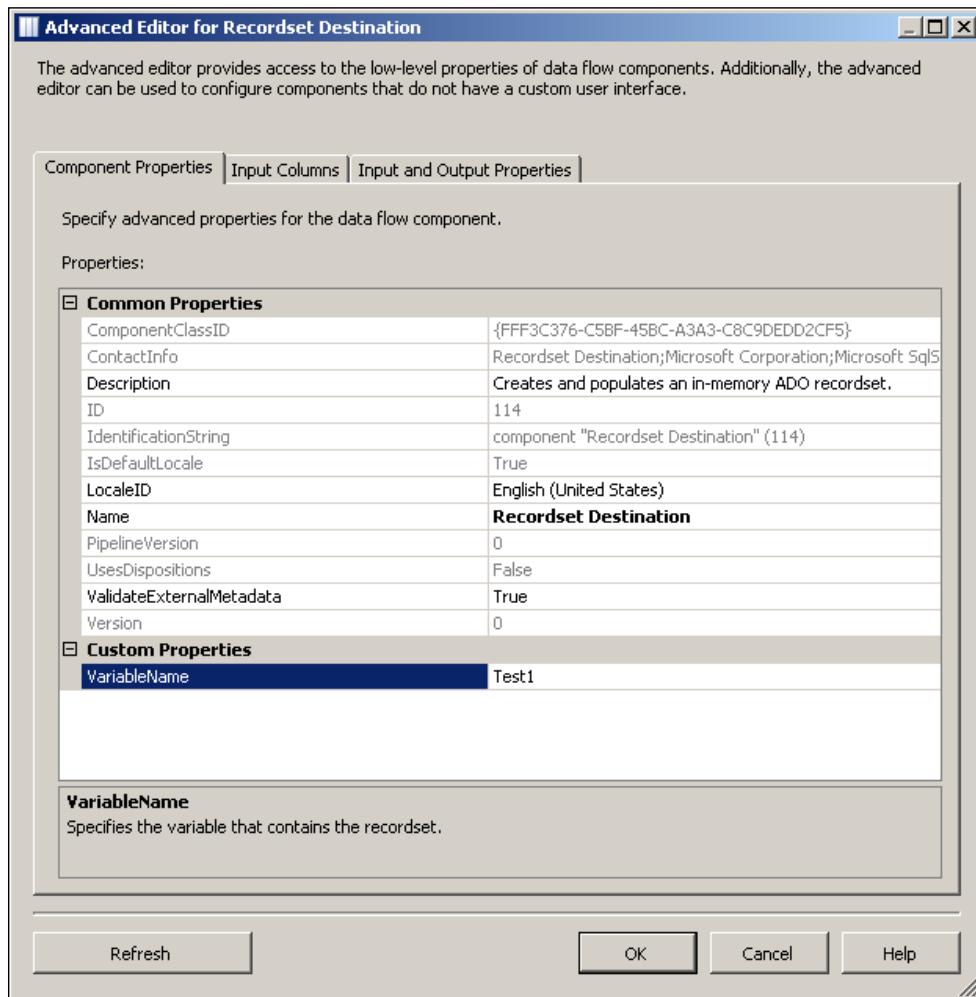
Step 6: Configure the Recordset Destination(s)

1. Right-click the **Recordset Destination** (the first one added) component in the canvas and choose the drop-down menu item **Variables**.
2. This opens the **Variables** window as shown in the next screenshot where we need to associate this recordset with a variable. A variable carries the information in the in-memory recordset and that is the reason we need to associate a variable with the recordset destination.

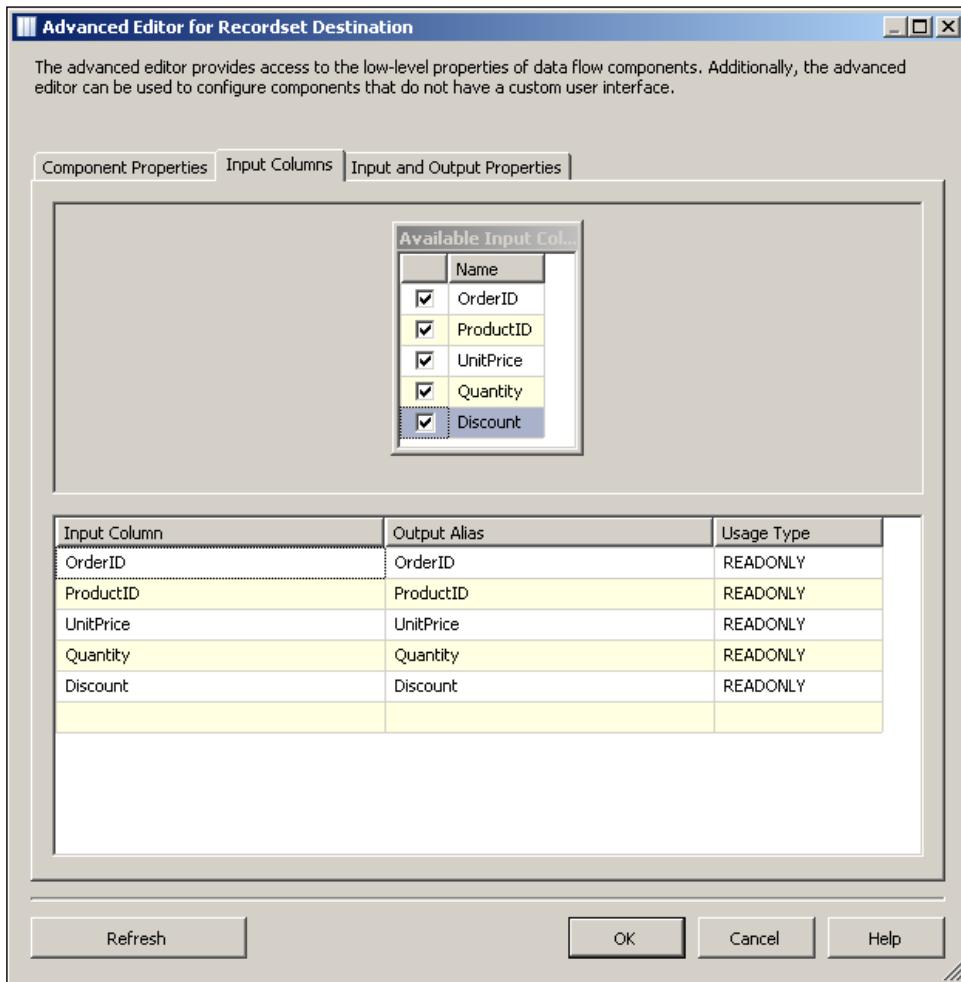


3. Click on the **Add** symbol in the **Variables** window.
4. In the **Variables** window, change the default values as follows: **variable** in **Name** to **Test1** [no space between Test and 1] by typing it in; **int32** to **Object** (choose from drop-down menu) since recordset's data type is object in **Scope**. The **Data Type** will display **Object**, and **Value** will display **System.Object**. A completed view of this window is presented later.
5. Now right-click the **Recordset Destination** and choose **Edit....**

This opens the **Advanced Editor for Recordset Destination** as shown in the following screenshot. For **VariableName** line item type in **Test1** by its side, as shown. If you did not configure a variable and try to proceed to the next tab you will be faced with an error.



6. Click on the **Input Columns** tab to open the **Input Columns** window. Place a check mark for the columns that you want into the recordset (all are chosen for this exercise) as shown in the following screenshot. Click on the **Refresh** button to refresh the data.
7. If you want, you may choose fewer columns as well as their output aliases.



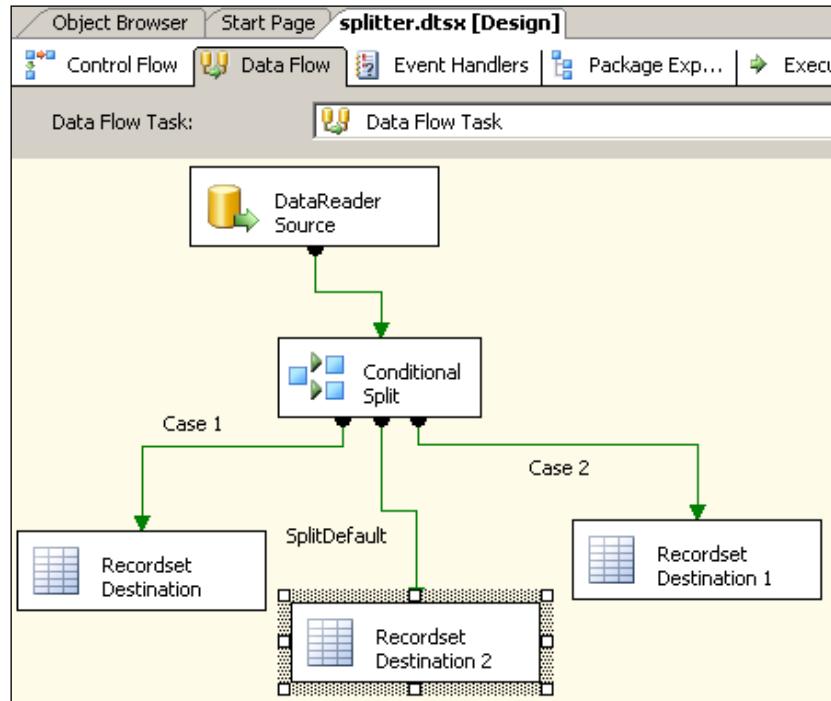
8. Click on the **OK** button because this type of destination does not have an output and no need to go to the next tab, **Input and Output Properties**.

This completes the configuration of the Recordset Destination that will have the data channelled by Case 1.

Using a Conditional Split Data Transformation

9. Repeat the same procedure for the other two outputs from the Conditional Split, **Case 2** and **SplitDefault**.

The completed design in the canvas should appear as shown in the following screenshot.

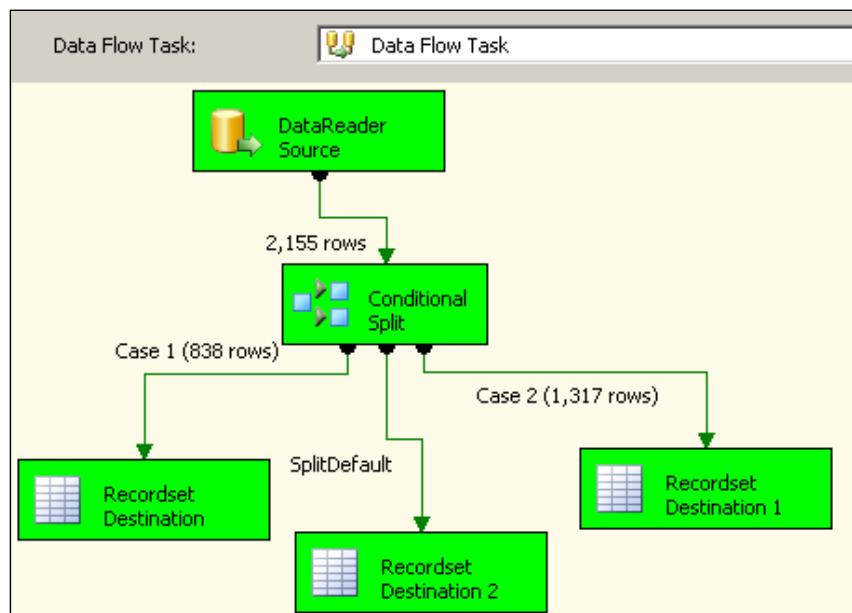


The associated **Variables** window with this task should appear as in the following screenshot.

Name	Scope	Data Type	Value
Test1	Data Flow Task	Object	System....
Test2	Data Flow Task	Object	System....
SplitDefault	Data Flow Task	Object	System....

Step 7: Build, Execute the Package and Review

Build the project and execute the package, similarly to how you did in other chapters. The program runs, and after a while you will see that all the objects turn green, indicating success as shown in the following screenshot. You will also see that 828 records went to the Recordset Destination for Case 1, and 1317 records went over to Case 2 making a total of 2155 rows. None went to the default. You may verify whether this is correct by running suitable select queries [for example, select count(*) from [Order Details] where Discount > 0] in the SQL Server.



Summary

This chapter described in detail the usage of a Conditional Split data transformation. You should use this generic task wherever you want to branch out based on some business logic. This logic can be something that is a part of the data itself, or some external conditions. Although the split data was channeled to three Recordset Destinations, one may very well use other types of destinations as well. In further chapters, we will see how the variable used in the Recordset Destination transformation can be used (at present it is showing the number of rows that are held therein).

9

Using an Aggregate Data Transformation

This chapter shows you how to create a package that can aggregate data for a given group of items in a Data Flow using the Aggregate Data Flow Transformation. You will also learn how to use the Percentage Sampling Data Flow Component. The data will be extracted from SQL Server 2005 and loaded to an in-memory Recordset Destination.

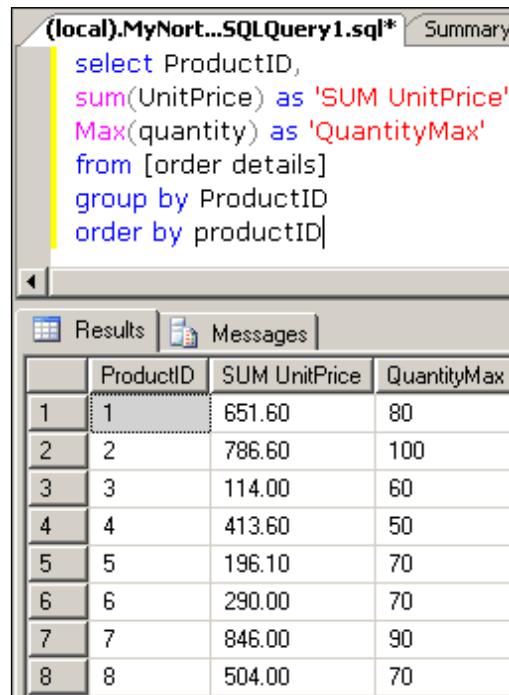
Structured Query Language's SELECT query is perhaps one of the most important statements that helps in identifying a single piece of information from a large database. In developing enterprise reports, there are often times you need to know consolidated specific pieces of information. For example, as a hiring manager you may be interested in knowing the average and minimum salaries you have to pay in a given geographical region for hiring. SQL provides the aggregate functions to address those kinds of questions to the database by working on multiple records of data. Using aggregate functions, you can find sum, maximum, minimum, average values of items in a table or a group.

The aggregate functions Count, Sum, Average, Min, and Max are all defined according ANSI SQL-92 standard. This standard also describes how nulls are handled while aggregating. The Aggregate Data Flow Transformation does the aggregation of data in the context of SSIS.

Hands-On Exercise: Using Aggregate Data Flow Transformation

In order to follow the steps as indicated, you will need a data flow task that connects to a data source and a Recordset Destination to which the data can flow. You will introduce an aggregate data flow component and a percentage sampling data flow item in the path of the data.

In this exercise, you will be aggregating data from the **OrderDetails** table discussed in Chapter 6. The next screenshot shows how an aggregate query is posed and the result of running this query in SQL Server 2005's Management Studio. The aggregate values (**SUM UnitPrice** and **QuantityMax**) are grouped by **OrderID**.



The screenshot shows a SQL query window titled '(local).MyNort...SQLQuery1.sql*'. The query is:

```
select ProductID,
       sum(UnitPrice) as 'SUM UnitPrice'
      Max(quantity) as 'QuantityMax'
     from [order details]
    group by ProductID
   order by productID
```

The results grid has three columns: ProductID, SUM UnitPrice, and QuantityMax. The data is as follows:

	ProductID	SUM UnitPrice	QuantityMax
1	1	651.60	80
2	2	786.60	100
3	3	114.00	60
4	4	413.60	50
5	5	196.10	70
6	6	290.00	70
7	7	846.00	90
8	8	504.00	70

The following are the major steps in this exercise:

- Create a BI Project and add a Data Flow Task. Add and configure the DataReader Source to extract data from the Local SQL Server.
- Add an Aggregate Data Transformation
- Establish a path to connect DataReader Source with the Aggregate Data Transformation.
- Configure the Aggregate Data Flow Transformation.
- Add a Percentage Sampling Data Transformation.
- Establish a path from Aggregate Data Transformation to the Percentage Sampling Data Transformation.
- Configure the Percentage Sampling Data Flow Item.
- Add a Recordset Destination Data Flow component.
- Configure the Recordset Destination Data Flow Component.
- Build and execute the package, and review results.

Step 1: Create a BI Project and Add a Data Flow Task. Add and Configure the DataReader Source to Pull Data from the Local SQL Server

1. Create a BI project Ch 9, change the name of the default Package to aggregate.dtsx, add a DataReader Source and configure it to provide at its output, data selected from the database with the following SQL statement:
Select * from [Order Details] order by ProductID

(The output of this query without the sort is shown in Chapter 6.)

Step 2: Add an Aggregate Data Transformation

The data output from the DataReader Source will be the input to the aggregate transformation. In the aggregate transformation's editor, the aggregation details will be set.

1. Drag and drop an **Aggregate Data Flow** item from the **Data Flow Transformations** group in the **Toolbox** to the **Data Flow** page of the canvas.

Step 3: Establish a Path to Connect DataReader Source with the Aggregate Data Transformation

1. Right-click the **Data Reader Source** and from the drop-down click on **Add Path**.

This displays the **Data Flow** window with the "From:" showing **Data Source Reader**. The process of establishing a path is same as in the previous chapters.

2. After choosing **Aggregate** in the "To:" drop-down, click on the **OK** button in the **Data Flow** window.

This opens the Input/Output Selection window where the **input**, **Aggregate Input1** is displayed.

3. Click on the drop-down arrow for choosing the "Output:". From the list, choose **DataReader Source**. Now, click on the **OK** button.

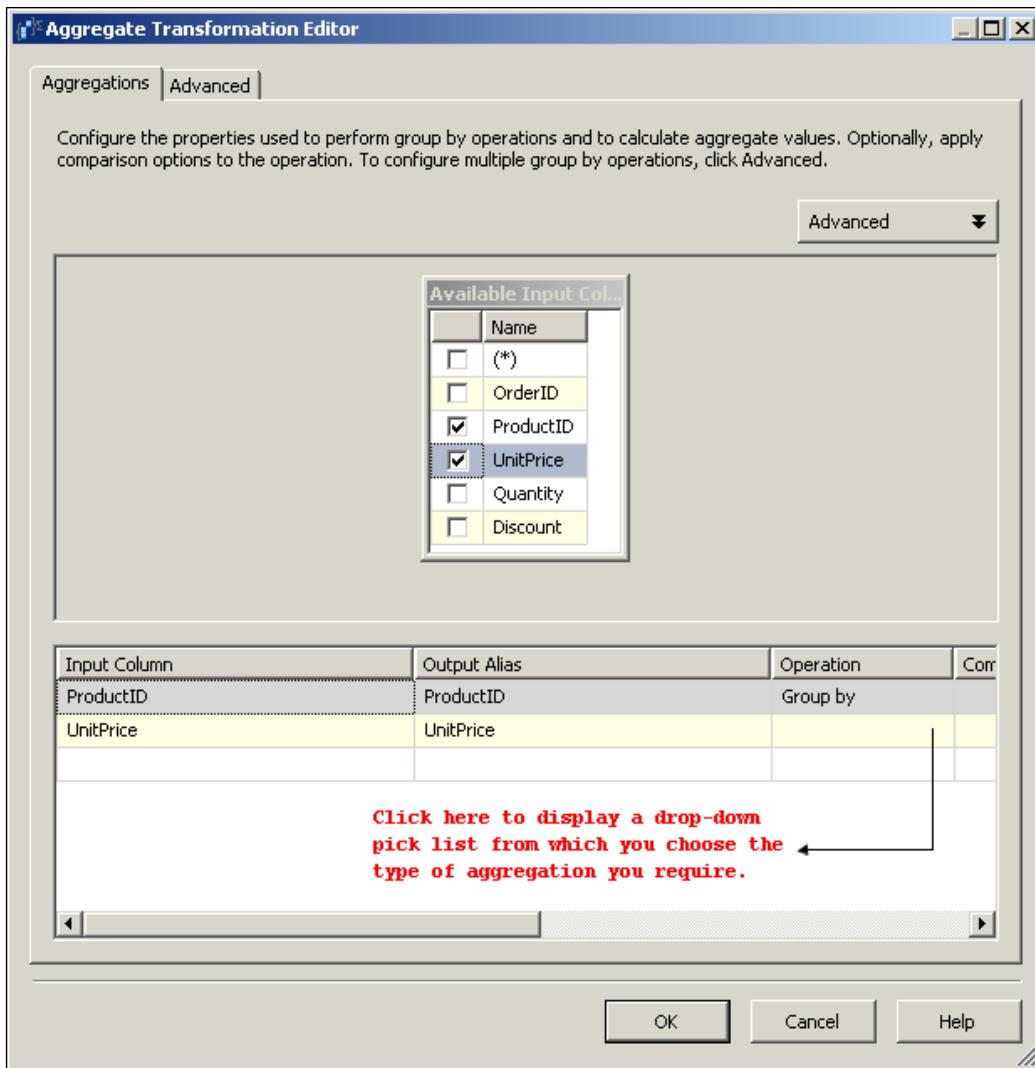
This establishes the path from the **DataReader Source** to the **Aggregate Data Flow** item. You should be able to see a thin green line from the **Data Reader Source** to the **Aggregate Data Flow Transformation** in the canvas. The input to the **Aggregate Data Flow** component is established.

Step 4: Configure the Aggregate Data Flow Transformation

1. Right-click the **Aggregate Data Flow** item and from the drop-down choose **Edit....**

This opens up the **Aggregate Transformation Editor** with two panes, a top pane showing all the columns from the **DataReader Source** output and a bottom pane with a list. In the top pane, all columns are unchecked. We are going to group the **ProductID**, and in that group we sum the **UnitPrice**. In order to do this:

2. Place check marks on the **ProductID** and **UnitPrice** columns in the top pane of the **Aggregate Transformation Editor**, as shown in the next screenshot.

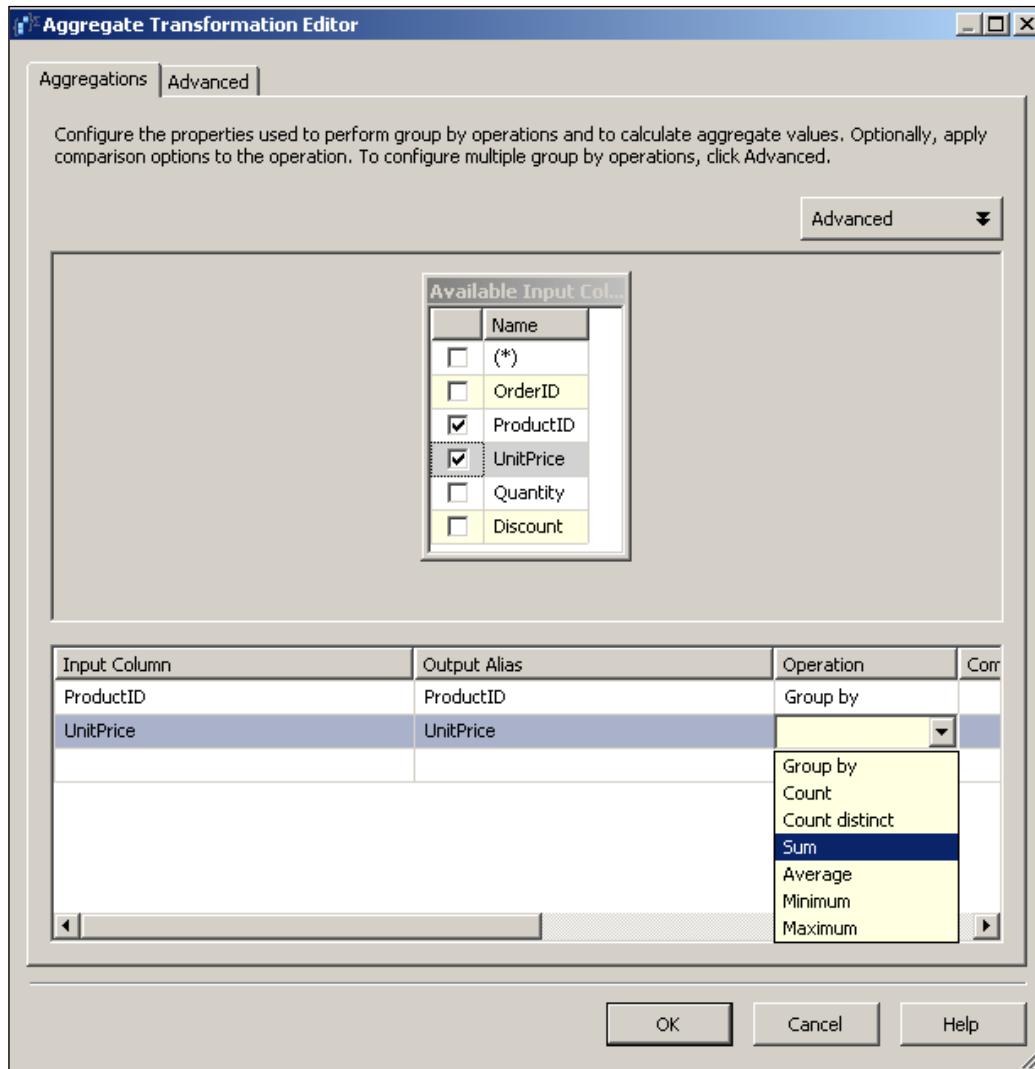


This adds the two line items to the list in the bottom pane, as shown in the previous screenshot. By default the Aggregate Transformation Editor already groups ProductID when the ProductID is checked.

Using an Aggregate Data Transformation

3. Click in the indicated position under the column heading **Operation**.

This opens the pick list as shown. You may choose any of the items in the list. For this exercise, the aggregate function **SUM** will be used.



4. Pick **SUM** from the list and click the **OK** button on this editor.

The **Aggregate Data Flow** item output is now available. The **Aggregate Data Flow** item can be used to configure multiple aggregate functions using the **Advanced** tab of this editor. For this exercise only, the basic editor functionality has been used.

Step 5: Add a Percentage Sampling Data Transformation

The Percentage Data Flow component was described in [Chapter 1](#). This component just randomly selects a preset percentage of rows of data to pass from its input to its output path. This component is useful in data mining and wherever you need a smaller representative set of data from a larger set, for example for testing.

1. Drag and drop a **Percentage Sampling Data Flow** item from the **Data Flow Transformations** group in the **Toolbox** to the **Data Flow** page of the canvas.

This adds the **Percentage Sampling Data Flow** item to the **Data Flow** page of the canvas.

Step 6: Establish a Path from Aggregate Data Transformation to the Percentage Sampling Data Transformation

1. Right-click the **Aggregate Data Flow** item and choose **Add Path**.

This opens the **Data Flow** window displaying the "From:" as an **Aggregate data flow** item.

2. Click on the drop-down arrow head in the "To:" window.

In the drop-down, you will see three options, **DataReader Source**, **Aggregate**, and **Percentage Sampling**.

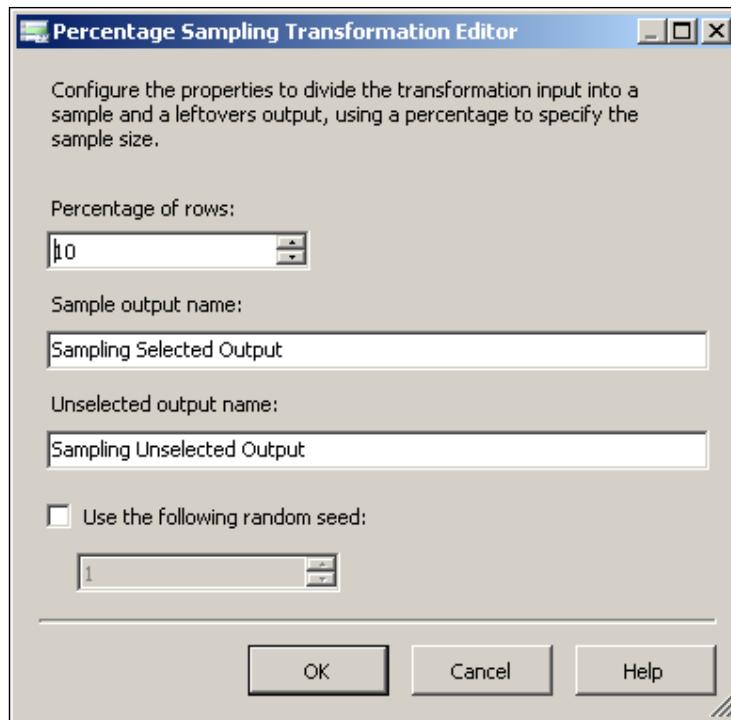
3. Choose **Percentage Sampling**. Click on the **OK** button in the **Data Flow** window.

This establishes the path from the **Aggregate Data Flow Item** to the **Percentage Sampling Data Flow** item, and you will be seeing a thin green line connecting the two.

Step 7: Configure the Percentage Sampling Data Flow Item

1. Right-click Percentage Sampling Data Flow item in the canvas and choose Edit....

This opens the Percentage Sampling Transformation Editor as shown in the next screenshot.



2. Read the instructions on this window. Change **Percentage of rows:** to 25 and use the default Random seed by checking the **Use the following random seed:** checkbox in the above window. Click on the OK button.

When you place a check mark, a little message window opens below the random seed drop-down describing under what conditions it will be desirable to use a random seed. You will be using the default. This completes the editing of the Percentage Sampling Data Flow item.

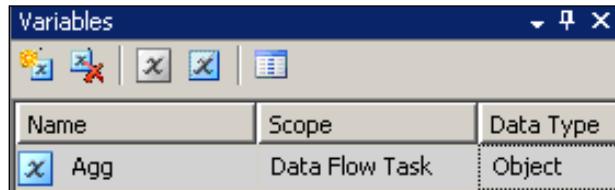
Step 8: Add a Recordset Destination Data Flow Component

We will add a **Recordset Destination Data Flow** component to the canvas and connect the output of the **Percentage Sampling Data Flow** item to the input of the **Recordset Destination**.

1. Drag and drop the **Recordset Destination** data flow component from the **Data Flow Destinations** group on to the canvas.
2. Right-click on the **Percentage Sampling Data Flow** item and from the drop-down choose **Add Path**.
3. Connect the Percentage Sampling Component's "Selected Sampling Output" to the **Recordset Destination**.

Step 9: Configure the Recordset Destination Data Flow Component

1. Right-click the **Recordset Destination Data Flow** component and from the drop-down list, click on variables to open the variables window.
2. In the variables window, click on **Add Variable**. Type in, or choose items as shown in the next screenshot (for details refer to the previous chapters).



3. Right-click the **Recordset Destination Data Flow** component and from the drop-down choose **Edit....**

This opens the **Advanced Editor for Recordset Destination**. In the **Component Properties** page, associate the above added variable **Agg** with the **VariableName** in the **Custom Properties** node, as shown in the next screenshot.

Properties:	
Common Properties	
ComponentClassID	{FFF3C376-C5BF-45BC-A3A3-C8C9DEDD2CF5}
ContactInfo	Recordset Destination;Microsoft Corporation;Microsoft SQL Server
Description	Creates and populates an in-memory ADO recordset.
ID	968
IdentificationString	component "Recordset Destination" (968)
IsDefaultLocale	True
LocaleID	English (United States)
Name	Recordset Destination
PipelineVersion	0
UsesDispositions	False
ValidateExternalMetadata	True
Version	0
Custom Properties	
VariableName	Agg

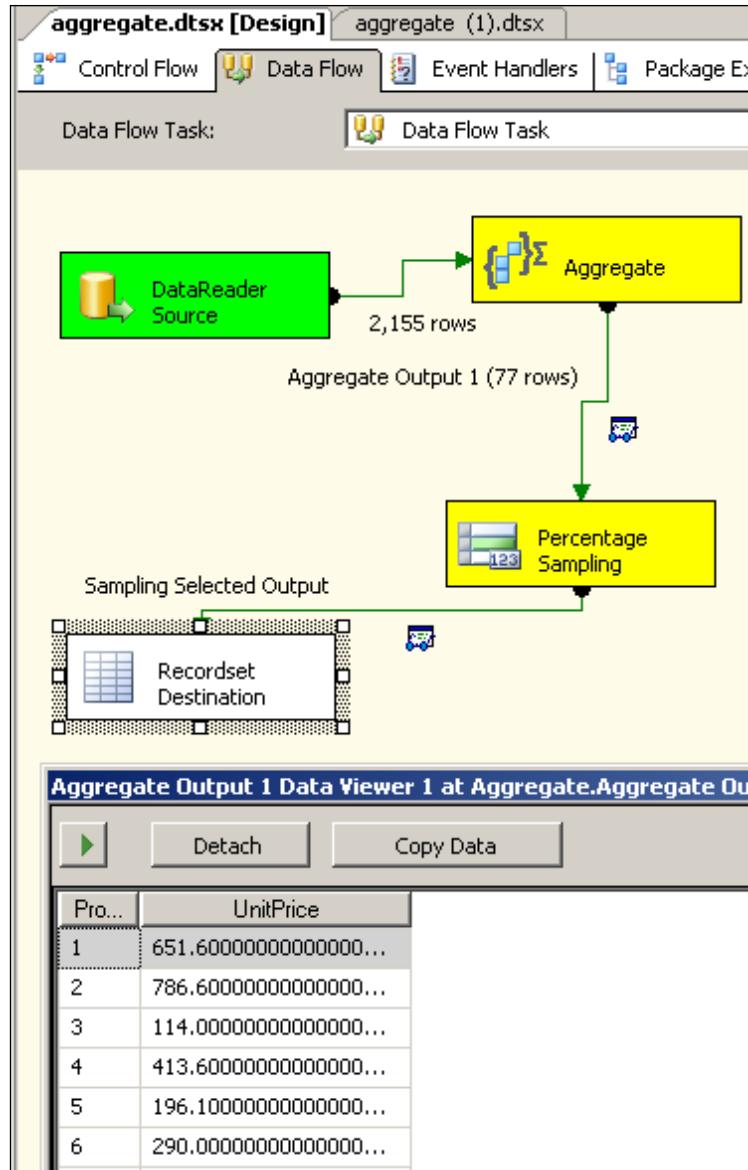
4. Click on the **Input Columns** tabbed page and you should see both the ProductID and the UnitPrice selected in a two pane window.

If needed, you may change the Output aliases. If you do not, the default will be used. For this exercise, the Output alias for the **UnitPrice** column is chosen as, **Sum [Unit Price]** and **Product ID** for ProductID.

5. Click on the **OK** button on the **Advanced Editor for Recordset Destination**. This completes the Recordset Destination editor configuration.
6. Right-click on the path from **Aggregate Data Flow** item to the **Percentage Sampling Data** item and add a **Grid** type **Data Viewer** as described in a Chapter 6.
7. Similarly, add a **Grid** type **Data Viewer** to the path from **Percentage Sampling Data Flow** item to the **Recordset Destination**.

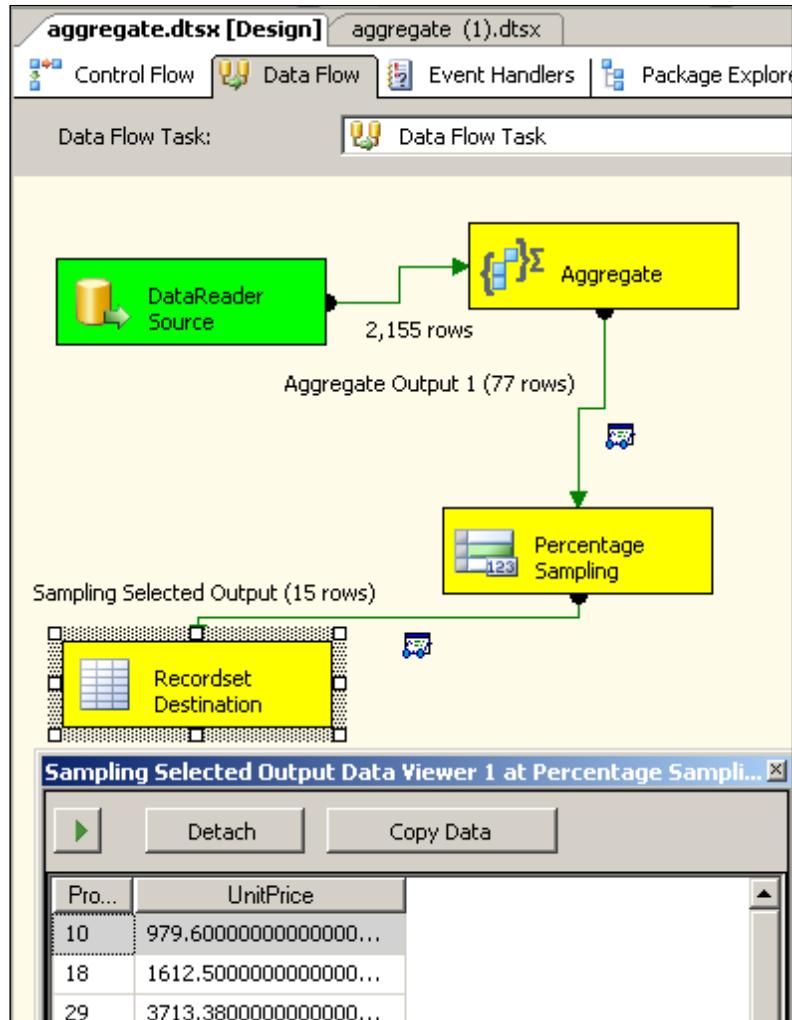
Step 10: Build and Execute the Package, and Review Results

Build the project and execute the package. The program runs and after a while you will see that the DataReader Source turns green while **Aggregate** and **Percentage Sampling** are yellow. 2155 rows transferred to the Aggregate which has produced 77 grouped outputs. The Aggregate output is displayed in the following **Data Viewer** shown next (only part of it is shown).



1. Click on the green arrow head in the **Aggregate Output 1 Data Viewer1 at Aggregate.Aggregate Output 1** window.

This opens the path for the second data viewer as shown in the next screenshot, wherein 15 rows are passed from the **Percentage Sampling** component to the **Recordset Destination**. The number of rows that are picked by the **Percentage Sampling** depends on a random algorithm.



Summary

This chapter described details of a package with an **Aggregate Data Transformation** followed by a **Percentage Sampling** component. Recordset Destination, together with data viewers, provided visual confirmation of the success of this package. The Percentage Sampling (15 rows is not 25% of 77 in this present example) may not provide an exact percentage of the input as it uses special algorithms for random data selection. Interested readers may refer to the SQL Server Books online. The **Advanced Editor** for the **Aggregate Data Flow** item has more advanced features that were not explored in this exercise.

10

Using a Data Conversion Data Flow Transformation

Migration of data from one system to another is always fraught with lots of problems. As sources (older generation in general) and destinations (newer, cutting edge) are neither: on the same kind of database, nor on similar kind of platforms, data conversion can become an arduous effort. Data quality at the source can be poor for any number of reasons, misspelled, wrong type of data in a field, missing data, etc. These are handled at the transformation stage of the ETL process by resorting to changing the attributes of the data and other cleansing procedures.

Knowledge of data structures of source and destination, data mapping, and the conversion process are all important for a successful data conversion operation. SSIS has built in tool to handle this process, *Data Conversion*. *Data Transformation* is therefore one of the main pieces in ETL.

In this exercise, we will extract a small table residing in an MS Excel spreadsheet file and transform this file before loading to an OLE DB Destination (an MDB file). The schema at the destination and the parameters chosen for data conversion are such that some data conversion errors are likely to be generated. The unacceptable data is routed to a recordset destination. In doing this exercise, you will learn the usage of an Excel Data Source besides configuring the Data Conversion Data transformation. You will also learn how to handle rows that cannot be loaded to the destination.

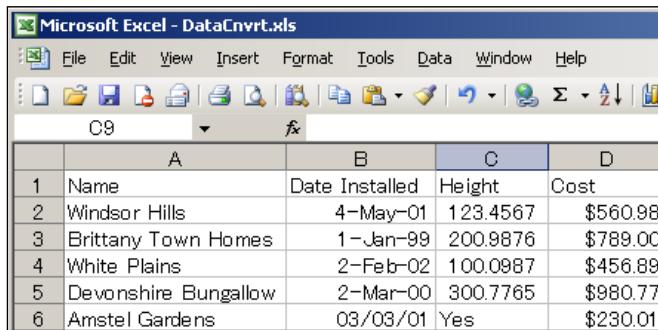
Source and Destination for the Exercise

The Excel source and the MS Access table will be described. As a preparation for this exercise, you should construct these simple files. This exercise will be using these files in the data conversion process.

- **Excel Source**

1. Create a MS Excel 2003 Spreadsheet file in c:\ drive, DataCnvt.xls, with entries as shown in the following screenshot and rename the sheet as **DataConvert**.

It has four different data types with an erroneous entry in one of its rows. This file is created in c:\ drive for convenience.

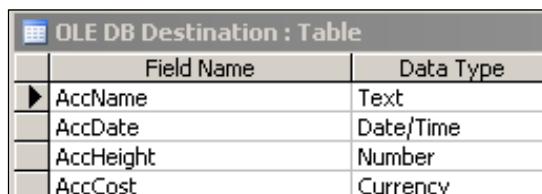


	A	B	C	D
1	Name	Date Installed	Height	Cost
2	Windsor Hills	4-May-01	123.4567	\$560.98
3	Brittany Town Homes	1-Jan-99	200.9876	\$789.00
4	White Plains	2-Feb-02	100.0987	\$456.89
5	Devonshire Bungalow	2-Mar-00	300.7765	\$980.77
6	Amstel Gardens	03/03/01	Yes	\$230.01

The data types in Excel columns are based on the OLE DB Specifications such as OleDbType.double for Numeric data like in the **Height** column; OleDb.Currency for the **Cost** column, etc. These can be determined by calling the appropriate OLE DB provider programmatically. For details, the KB article (<http://support.microsoft.com/kb/318373/en-us>) may be useful.

- **OLE DB Data Destination**

1. Create a MS Access application DataConvert.mdb. In this database, create a table, **Access Destination**, having the following design. This table will be empty to start with, but will be filled with data from the XLS file after the data conversion. The data types are the ones that are available in MS Access 2003.



Field Name	Data Type
AccName	Text
AccDate	Date/Time
AccHeight	Number
AccCost	Currency

Set the attributes of each of the columns as shown here (these were arbitrarily chosen, Field Size for **AccName** was chosen to generate error):

- **AccName** – Data Type: text; Field Size: 15; Allow Zero Length:No
- **AccDate** – Data Type: Date/Time; Format; General Date

- **AccHeight** – Data Type: Number, Field Size: Single; Decimal Places:2
 - **AccCost** – Data Type: Number; Field Size: Long Integer; Format Currency, Decimal Places: 3; Default Value: 0

None of these are required. None of these are indexed.

Hands-On Exercise: Transferring Data to an Excel File

This exercise consists of the following major steps:

- Creating a BI project; adding a Data Flow Task and configuring an Excel Source.
- Adding a Data Conversion Data Flow Transformation.
- Establishing a path from Excel Source to Data Conversion Data Flow Transformation.
- Configuring the Data Conversion Data Flow Transformation.
- Adding and configuring an OLE DB Data Destination.
- Adding and Configuring a Recordset Destination for displaying errors.
- Building the project and testing the package.

Step 1: Creating a BI Project; Adding a Data Flow Task and Configuring an Excel Source

Create a BI project Ch 10. Change the name of the default package to DCnvrt.dtsx. Drag and drop a **Data Flow Task** to the **Control Flow Page**.

1. Drag and drop an **Excel Source** item from the **Data Flow Sources** group in the **Toolbox** to the **Data Flow Page**.

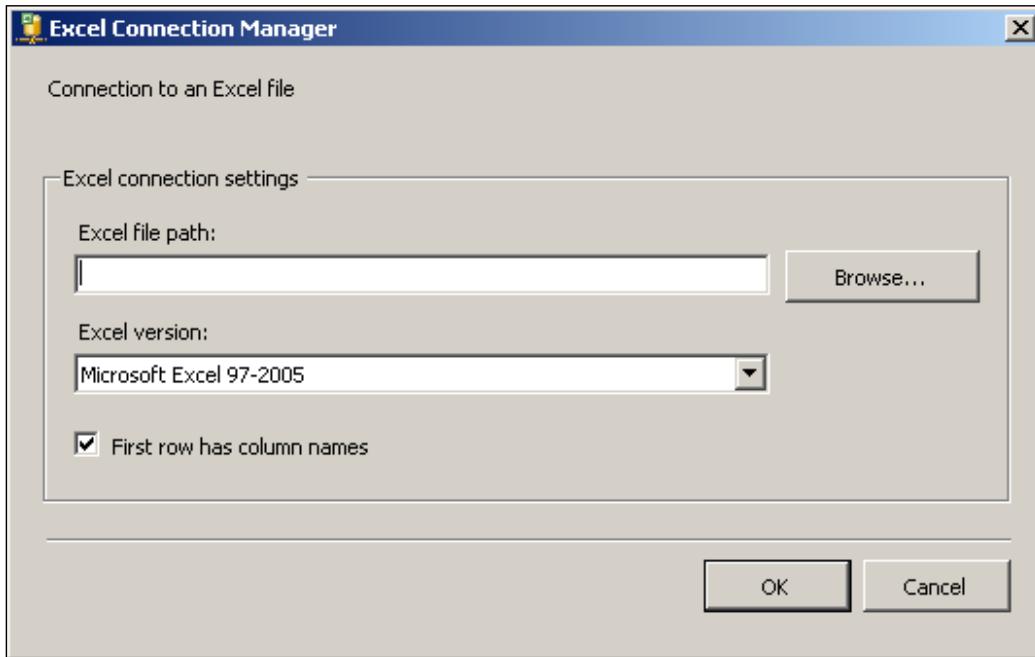
An instance of the Excel Source will be added to the DCnvrt.dtsx package.

2. Right-click the **Excel Source** item, and from the drop-down menu choose **Edit....**

This brings up the **Excel Source Editor** window.

3. In the Excel Source Editor that shows up, click on the **New...** button.

This opens the **Excel Connection Manager** page as shown below.



4. Click on the **Browse...** button and locate the `DataCnvrt.xls` file and choose the file.

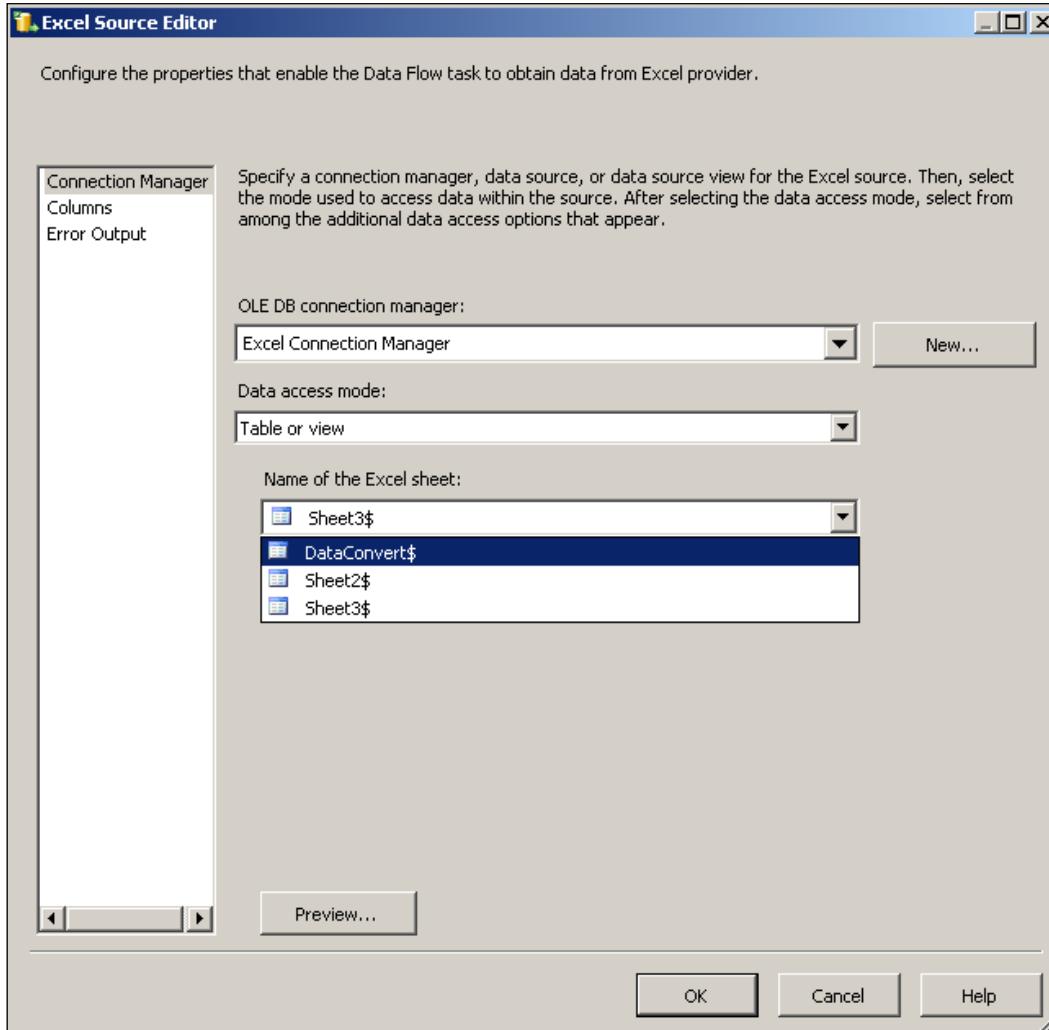
This will bring the file to the **Excel file path:** field, shown in the previous screenshot.

5. Click **OK** once the file name is displayed.

This will take you back to the **Excel Source Editor**. The **OLEDB connection manager:** field gets an entry -"Excel Connection Manager". An icon will be added to the **Connection Managers**' page. The **Name of the Excel Sheet:** field is now enabled.

6. Click on the **Name of Excel Sheet:** drop-down and choose **DataConvert\$**, as shown in the following screenshot.

You may click on the **Preview...** button to see how the sheet looks. The *preview* runs a query and shows the original Excel sheet you started during the preparatory step.

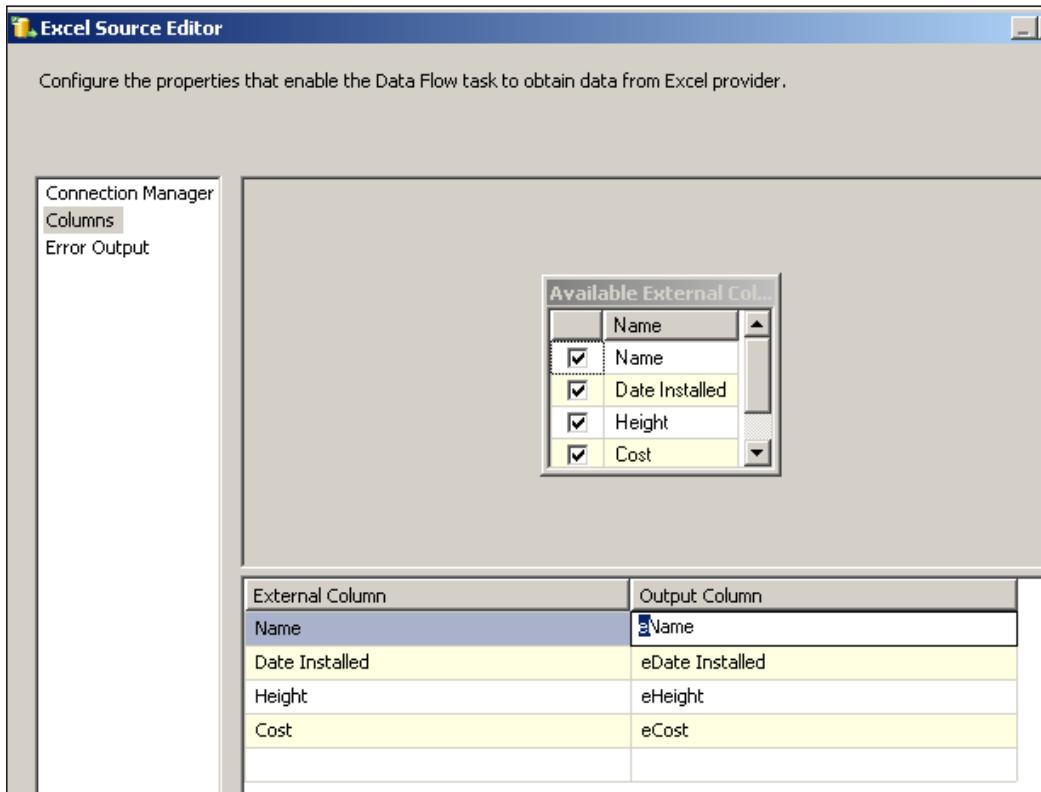


7. Click on **Columns**, shown in the left-hand side of the above screenshot.

This opens the Available External Columns and the Output Columns page, as shown in the next screenshot.

- Alter the **Output Column** rows by adding a letter "e" to the beginning of each row. For example **Name** is changed to **eName** etc.

At present, we will not configure the error output as this is a simple contrived example that will not produce any error.



- Click on the **OK** button in the above window to complete the Excel Source configuration.

Step 2: Adding a Data Conversion Data Flow Transformation

- Drag and drop a **Data Conversion** item from the **Data Flow Transformations** group in the **Toolbox** onto the **Data Flow Page** of the Canvas.

This adds a Data Flow item to the **Data Flow Page** of the Canvas.

Step 3: Establishing a Path from Excel Source to Data Conversion Data Flow Transformation

1. Right-click the **Excel Source** and from the drop-down choose **Add Path**.
2. Establish a path from **Excel Source** component to the **Data Conversion** component using the interactive path connection GUI.

Complete the **Data Flow** path window that shows up so that the **From To:** points to **Excel Source** and the **TO:** label points to **Data Conversion**. Use the drop-down in the "To:" to make this happen.

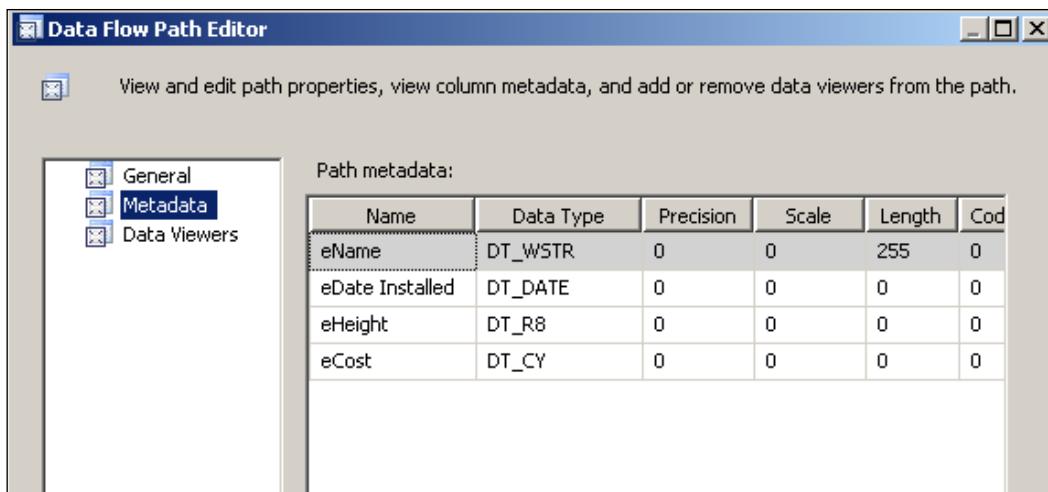
3. Click on the **OK** button and complete the **Input Output Selection** window that gets displayed so that the **Output:** points to **Excel Source Output** and the **Input:** points to **Data Conversion Input**.
4. Click on the **OK** button in the **Input Output Selection** window.

The **Excel Source** and the **Data Conversion** item are connected by a path, as indicated by a thin green line connecting them.

You may review how the data flows from **Excel Source** to the **Data Conversion** item by looking at how the path is configured.

5. Make a right-click on the path and choose **Edit....** This brings up the **Data Flow Path Editor** window. Click on **Metadata** on the left-hand side.

This displays the Metadata of the data in the path as shown in the following screenshot. The data types shown are SSIS defaults for the Excel Source.

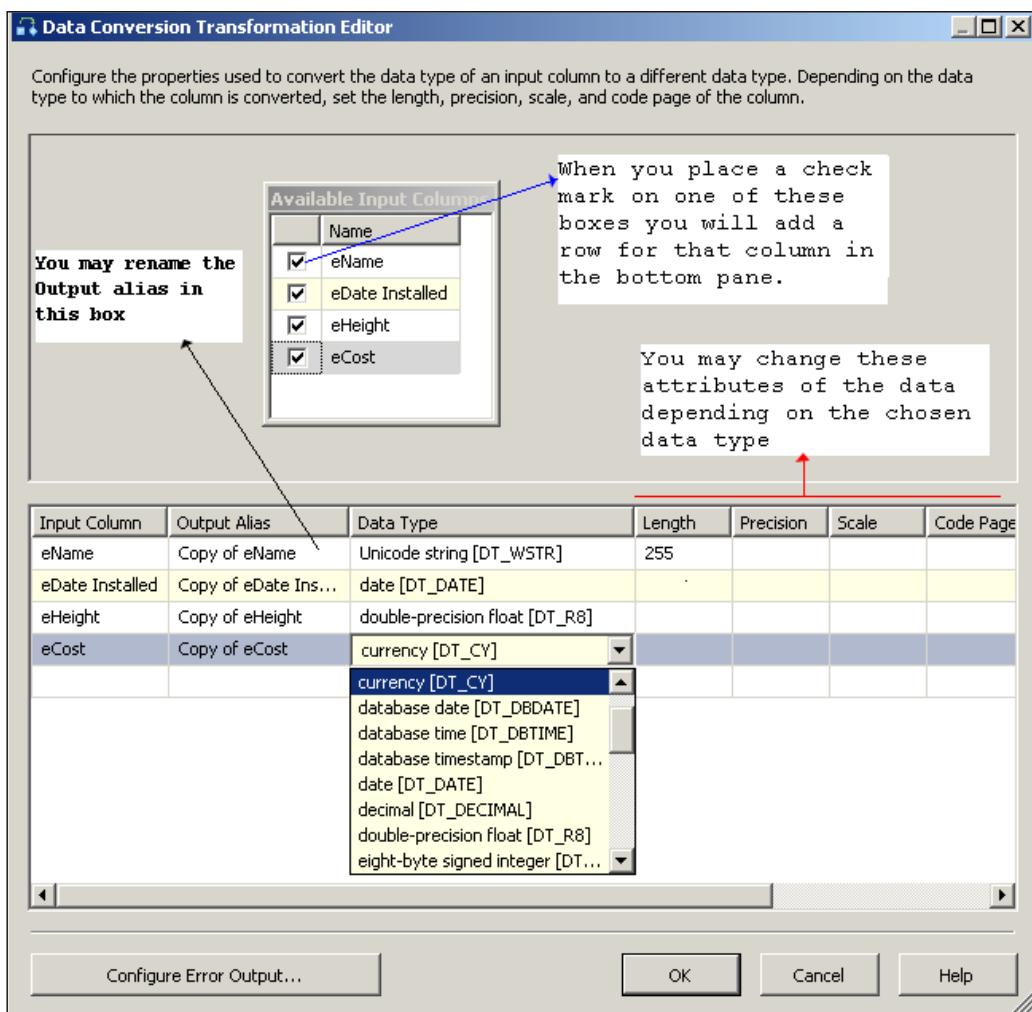


6. After reviewing the Metadata, click on the **Cancel** button.

Step 4: Configuring the Data Conversion Data Transformation

1. Right-click the Data Conversion item in the Canvas and choose **Edit...** from the drop-down menu.

This opens the **Data Conversion Transformation Editor** as shown in the following screenshot. When it gets displayed, the bottom pane will be empty and the table columns in the top pane will be unchecked. In the next mini-step, you will make changes as suggested in the Editor.



2. Place a check mark in all the columns. Change the **Output Alias** column names so that **Copy of eName** will be **dcName**, etc.

The **Data Type** that initially shows up is what SSIS has determined appropriate. If you go ahead with these values, you will probably, successfully make an error-free conversion.

3. Using drop-down controls change the **Data Type** to have the following values:

- **dcName:** Unicode string[DT_WSTR]
- **dcDate:** date[DT_DATE]
- **dcHeight:** Numeric Precision:18, Scale:3
- **dcCost:** Decimal[DT_DECIMAL]: Scale:3

The incoming data will be converted to the above data types after the transformation becomes effective. We will not use the **Configure Error Output...** option at present.

4. Click on the button **OK** in the **Data Conversion Transformation Editor** window after making the above changes.

Step 5: Adding and Configuring an OLE DB Data Destination

Adding an OLE DB Destination to Canvas

1. Drag and drop an **OLE DB Destination** item from the **Data Flow Destinations** group on to the **Data Flow** page of the Canvas.

Establish a Path from Data Conversion Data Flow Transformation to OLE DB Data Destination

1. Establish a path from the Data Conversion component to the OLE DB Destination component.
2. Right-click the **Data Conversion** object in the **Data Flow** page and choose the **Add Path** drop-down menu item, and configure the **Data Flow** window that shows up so that **From:** points to **Data Conversion** and **To:** points to **OLE DB Destination** using the drop-down handle.

3. When you click on the **OK** button (shown in the previous screenshot), you will display the **Input Output Selection** window. Configure the window so that the **Output:** points to **Data Conversion Output** and the **Input:** points to **OLE DB Destination Input** using the drop-down handle. After this, click on the **OK** button on the **Input Output Selection** window.

Now you will review the Metadata of this path segment from the Data Conversion component to the OLE DB Destination component.

4. Right-click this path segment and choose **Edit....** In the **Data Flow Path Editor** window, click on the **Metadata** on the left to display the following read-only window (only a part of this is shown).

This window shows both the incoming and outgoing data types using the Data Conversion Object.

Path metadata:					
Name	Data Type	Precision	Scale	Length	Cod
eName	DT_WSTR	0	0	255	0
eDate Installed	DT_DATE	0	0	0	0
eHeight	DT_R8	0	0	0	0
eCost	DT_CY	0	0	0	0
dcName	DT_WSTR	0	0	255	0
dcDate Installed	DT_DATE	0	0	0	0
dcHeight	DT_NUMERIC	18	3	0	0
dcCost	DT_DECIMAL	0	3	0	0

Setting Up a Connection Manager

1. Right-click the **OLE DB Destination** object and choose **Edit** to display the **OLE DB Destination Editor**.

In this window, you will see a list box displayed on the left with the following items: **Connection Manager**, **Mappings**, and **Error Output**. On the right, you will see only the **Data access mode** field displaying **Table or View**. The **OLEDB connect manager** and **Name of the table or view** fields are empty.

2. Click on the **New...** button alongside **OLE DB connection manager: label**.

This opens the **Configure OLE DB Connection Manager** window.

3. Click on the **New...** button on that window.

This will display the **Connection Manager** window.

4. Using the drop-down handle alongside **Provider:**, choose **Microsoft Jet 4.0 OLE DB Provider**.

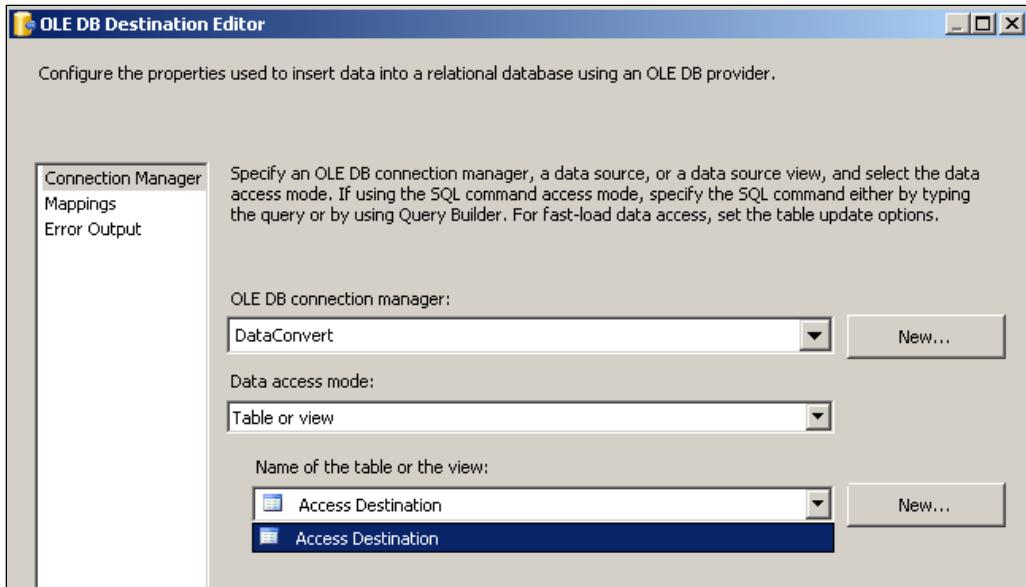
The items in the bottom of this screen changes to show required items for an MS Access database.

5. Click on the **Browse...** button and from C: drive choose the C:\DataConvert.mdb file. The **User name:** default is **Admin** and no need for a password as the present user is also the owner. You may also test the connectivity using the **Test Connection** button.
6. Click on the **OK** button after this, the C:\DataConvert.mdb will be added to the **Data connections:** list box on the left of the **Configure OLE DB Connection Manager's** window. Click on the **OK** button on this window.

This will display the **DataConvert** connection manager in the **OLE DB Destination Editor** window as well as adding an icon to the **Connection Manager's** window of the Canvas.

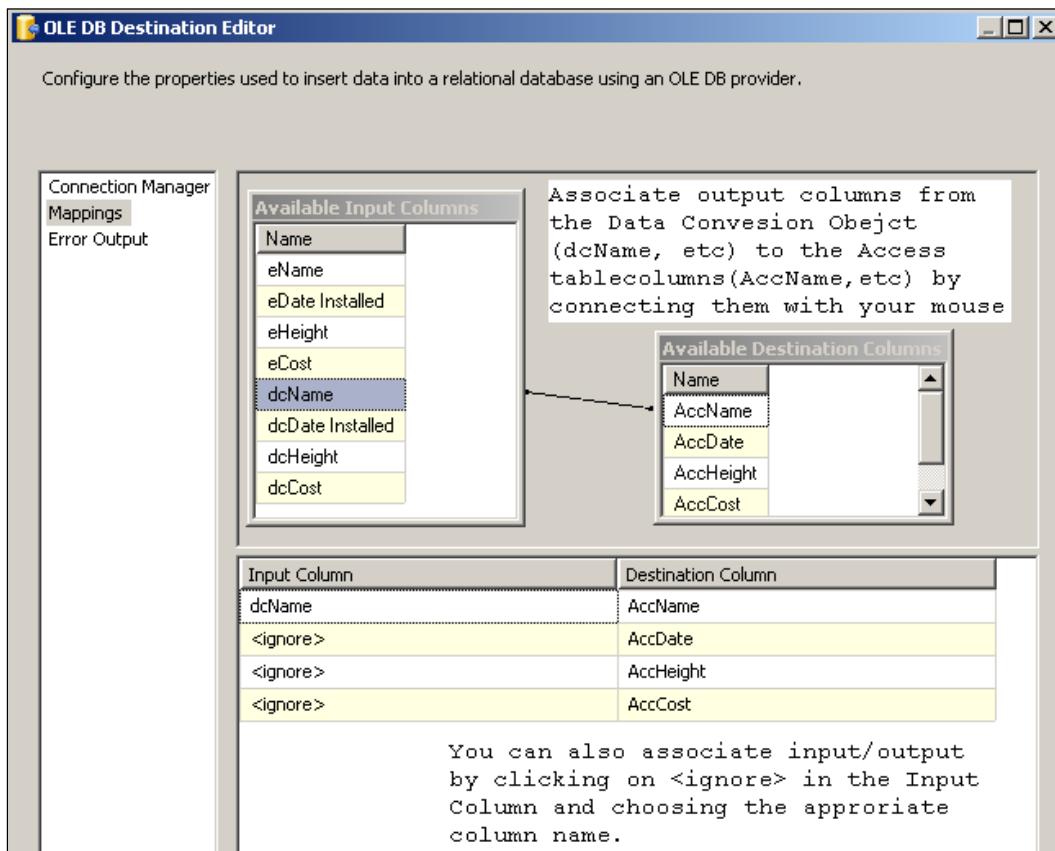
Displaying the Table to which the Data will be Inserted

Click on the drop-down along the **Name of the table or view:** label on the **OLE DB Destination Editor's** window and choose **Access Destination** table from the drop-down, as shown in the next screenshot. At this point, if you click on **Preview** button, you will see the table fields but no data.



Column Mappings

- Click on the **Mappings** list item on the left of the previous figure. Follow guidance in the figure and associate the input / output columns for all columns (only shown for one).

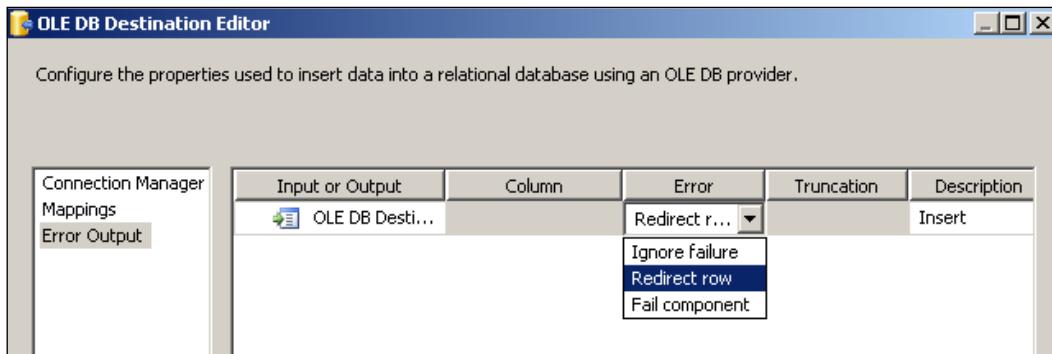


Handling Data Loading Errors

Looking at the `DataCnrvrt.xls` sheet entries and the MS Access Database table `AccessDestination` table column data types, it is easy to see that there will be errors in loading the data. In this section, we will configure the errors to be ported to a **Recordset Destination** item.

- Click on the **Error Output** list item on the left. This opens the window shown in the next screenshot where you can indicate how errors are to be handled.

There are three options (as shown in the screenshot) displayed in the drop-down list in the error column.



2. In the **Set this value to selected cells:**, click on drop-down and select **Redirect row**.

The rows that had errors will be redirected to the error output, the red line between components in the Canvas.

3. Click on the button **OK** on this window.

Step 6: Adding and Configuring a Recordset Destination for Displaying Errors

You add a **Recordset Destination** to monitor the error and details of errors. Adding a **Recordset Destination** has been described in earlier chapters.

1. Drag and drop a **Recordset Destination** from the **Toolbox**.
2. Drag the dangling red line from **OLE DB Destination** that represents the error and connect it to the **Recordset Destination**.
This establishes a path for the error to flow to the Recordset Destination.
3. Double-click the **Recordset Destination**.
This opens up the **Advanced Editor for Recordset Destination**.
4. In the Component Properties page, provide a **variable** (herein called DCerr) and associate the **Recordset Destination** with this variable.
5. Choose the input columns to the **Recordset Destination** in the tabbed page **Input Columns** of the **Advanced Editor for Recordset Destination**.

In addition, you add **Data Viewers** to the completed Canvas, one in the path of the OLE DB Destination and the other in the path of the Recordset Destination.

Step 7: Building the Project and Testing the Package

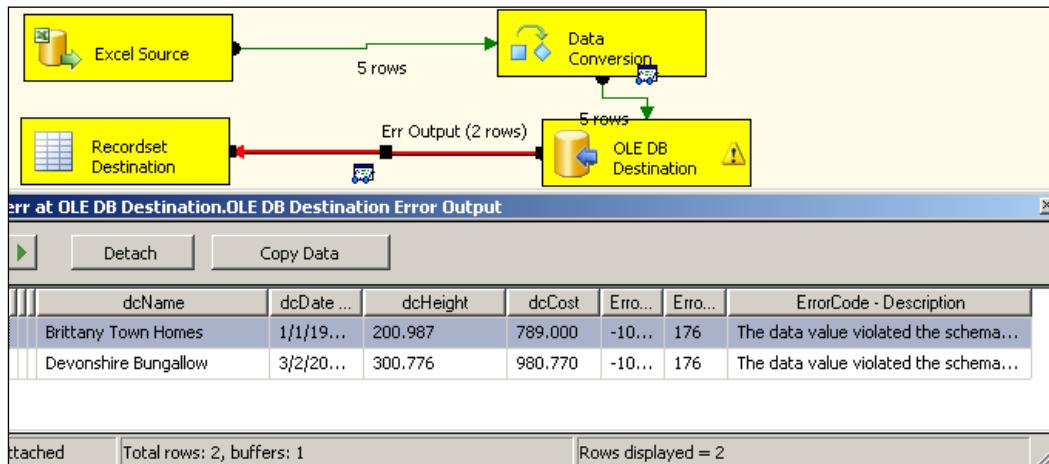
1. Build the project and execute the package as described in the previous chapters.

The program cranks up and you will see all the five rows passing from Data Conversion object to the OLE DB Destination object as shown in the following screenshot.

all at Data Conversion.Data Conversion Output								
eName	eDate Inst...	eHeight	eCost	dcName	dcDate ...	dcHeight	dcCost	
Windsor Hills	5/4/2001...	123....	560.98	Windsor Hills	5/4/20...	123.456	560.980	
Brittany Town Homes	1/1/1999...	200....	789	Brittany Town Homes	1/1/19...	200.987	789.000	
White Plains	2/2/2002...	100....	456.89	White Plains	2/2/20...	100.098	456.890	
Devonshire Bungalow	3/2/2000...	300....	980.77	Devonshire Bungalow	3/2/20...	300.776	980.770	
Amstel Gardens	3/3/2001...	NULL	230.01	Amstel Gardens	3/3/20...	NULL	230.010	

2. Click on the button with the green arrow head, shown in the above screenshot.

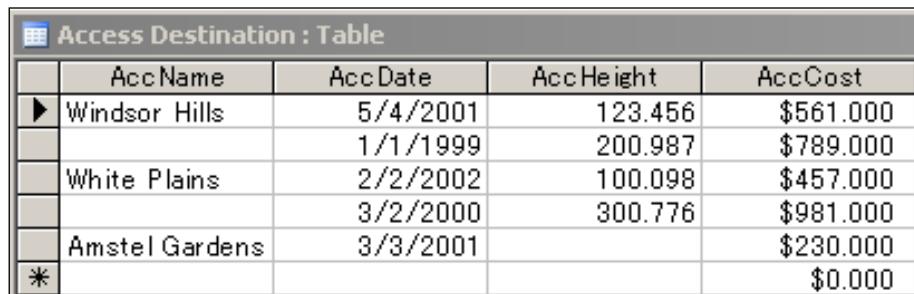
The rows with errors are redirected to the Recordset Destination as shown in the next screenshot. Also notice that the error messages that are in these rows. Two rows of incoming data violated the schema of the table into which they were scheduled to be loaded.



3. Click on the button with the green arrow head now.

This will turn all the components green.

4. Open the **Access Destination** table in the DataConvert.mdb access file to review the rows transferred. The next screenshot shows the **Access Destination** table in **Table View**.



	AccName	AccDate	AccHeight	AccCost
▶	Windsor Hills	5/4/2001	123.456	\$561.000
		1/1/1999	200.987	\$789.000
	White Plains	2/2/2002	100.098	\$457.000
		3/2/2000	300.776	\$981.000
	Amstel Gardens	3/3/2001		\$230.000
*				\$0.000

The two rows with missing **AccNames**, were the ones that had schema violation errors. Truncation errors such as this are not serious errors and therefore you still see a warning symbol on the OLE DB Destination component.

Summary

This chapter described in detail the steps involved in interposing a Data Conversion Data Transformation and how to guide the rows that violated to another location for review.

11

Creating a SSIS Package with an XML Task

XML, an acronym for Extensible Markup Language recommended by W3C (<http://www.w3c.org/xml/>), in 1998 has become the *lingua franca* of the internet. In the past ten years it has made tremendous inroads into all activities connected with the web and largely determines how data is exchanged between businesses. Because of the precise way data can be defined using a simple and flexible text format, it is efficient and best suited for data exchange across the internet. In this process, a number of other auxiliary constructs have grown to display (XSL), and query (XPath: XML Path language) XML documents. XMLTask in SSIS can carry out a number of tasks, and this chapter describes two of them in detail.

In this exercise, we will compare two XML documents using XMLTask, as simple comparison of text will not yield the differences correctly. We will also use XML task to convert an XML document to an HTML document containing the same data, but formatted in a prettier way to be viewed in a browser.

In XMLTask, you can carry out a number of tasks, which are discussed in the following sections:

Diff

When you want to compare two XML documents, and find differences between them, you can use this type. The result will be written to a third XML document called `XMDiffGram`.

The manner in which differences between two XML files are found is based on the operation type, `Diff`, in XMLTask. There are three ways of doing this—`Auto`, `Fast`, or `Precise`. `Precise` is accurate, and uses the **Zhang-Sasha** algorithm, where distance between trees in files is used, but is not a fast process. `Fast` is quick, but not precise.

It searches node-by-node to find the changes. Using `Auto`, the program chooses either `Precise` or `Fast`, depending on how the internals of XMLTask evaluate, assuming the size and extent of changes.

Merge

You can merge two XML documents and in fact you can even tell where in the first document you want the second to be merged.

Validate

Validating an XML document is important if it is going to be used in an application. You can validate it in two ways, using **Document Type Definition (DTD)**, or **XML Schema Document (XSD)**. XMLTask in SSIS can do it both ways.

XPATH

You can query an XML document using XPATH and evaluate the result.

XSLT

Rendered on a web page, an XML document is not pretty. An HTML page can be made pretty using tags, or CSS. You can convert an XML document to another format, say HTML, using this task for which you will need a document that can do this conversion for you. This conversion is the XSL transformation, which you define in another document called an XSL document.

In the two exercises, later in this chapter, we will be looking at **Diff** and **XSLT** task types of XML Task.

XML Documents Used in This Chapter

In this section, we will look at the different documents that we will be using, and where they are stored on the machine. The XML documents used in the exercise are files that have been used earlier (<http://hodentek.blogspot.com/2007/07/simple-xml-file-aka-my-hello-world-for.html>). Although the XML files used in this chapter are stored in the default directory of the local IIS, which is usually, `C:\Inetpub\wwwroot`, they can be stored in any directory and IIS.

Copy the XML documents (webstudentsOK.xml and webstudentsOK2.xml, and webstudents.xsl) and save them in any folder of your choice.

The results shown in this chapter, however, assume the files are stored in C:/inetpub/wwwroot

Documents Used for Diff XML Task Type

The webstudentsOK.xml document is an XML representation of data from a class consisting of 4 students; they have an ID, a name, and their skill(s) in programming. This is a well-formed document.

Listing of webstudentsOK.xml

```
<wclass>
  <!-- My students who attended my web programming class -->
  <student id="1">
    <name>Linda Jones</name>
    <legacySkill>Access, VB5.0</legacySkill>
  </student>
  <student id="2">
    <name>Adam Davidson</name>
    <legacySkill>Cobol, MainFrame</legacySkill>
  </student>
  <student id="3">
    <name>Charles Boyer</name>
    <legacySkill>HTML, Photoshop</legacySkill>
  </student>
  <student id="4">
    <name>Charles Mann</name>
    <legacySkill>Cobol, MainFrame</legacySkill>
  </student>
</wclass>
```

The webstudentsOK2.xml document has all the students of the first document, and it has an additional student, James Bond added to the end of the document, as shown in the following listing.

Listing of webstudetnsOK2.xml

```
<wclass>
  <!-- My students who attended my web programming class -->
  <student id="1">
    <name>Linda Jones</name>
    <legacySkill>Access, VB5.0</legacySkill>
  </student>
  <student id="2">
    <name>Adam Davidson</name>
```

```
<legacySkill>Cobol, MainFrame</legacySkill>
</student>
<student id="3">
<name>Charles Boyer</name>
<legacySkill>SSIS, SQL 2005</legacySkill>
</student>
<student id="4">
<name>Charles Mann</name>
<LegacySkill>Cobol, MainFrame</LegacySkill>
</student>
<student id="5">
<name>James Bond</name>
<legacySkill>Killer Apps</legacySkill>
</student>
</wclass>
```

Documents Used for XSLT XMLTask Type

This is same as the first document, `webstudentsOK.xml` under **Diff XMLTask type**. This is a well-formed XML document with `<wclass>` as root, containing four student nodes with **ID** as their attribute.

The XSL Document to transform the above XML is given as follows:

Listing of the `webstudents.xsl`

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:template match="/wclass/Student">
<HTML>
<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<xsl:value-of select="name"/>
<xsl:apply-templates select="legacySkill"/>
</BODY>
</HTML>
</xsl:template>
<xsl:template match="legacySkill">
<DIV>
studied
<span style="color:red; font-weight:bold">
<xsl:value-of select="."/>
</span>
</DIV>
</xsl:template>
</xsl:stylesheet>
```

Hands-On Exercise 1: XMLTask type Diff

In this exercise we will use the two documents, `webstudentsOK.xml` and `webstudentsOK2.xml`, and generate a `DiffGram` document that shows the difference between the two documents.

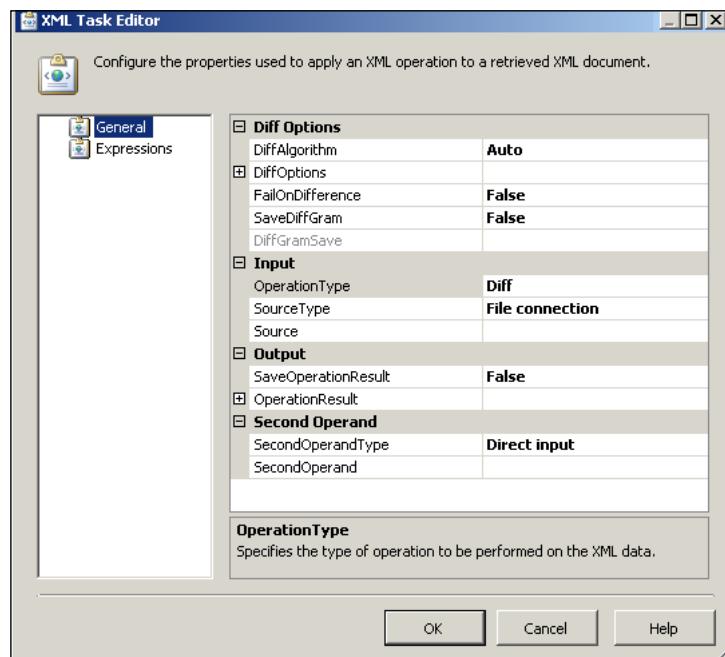
Step 1: Create a BI Project and Add a Control Flow task, XMLTask

This process has been described a number of times in the previous chapters.

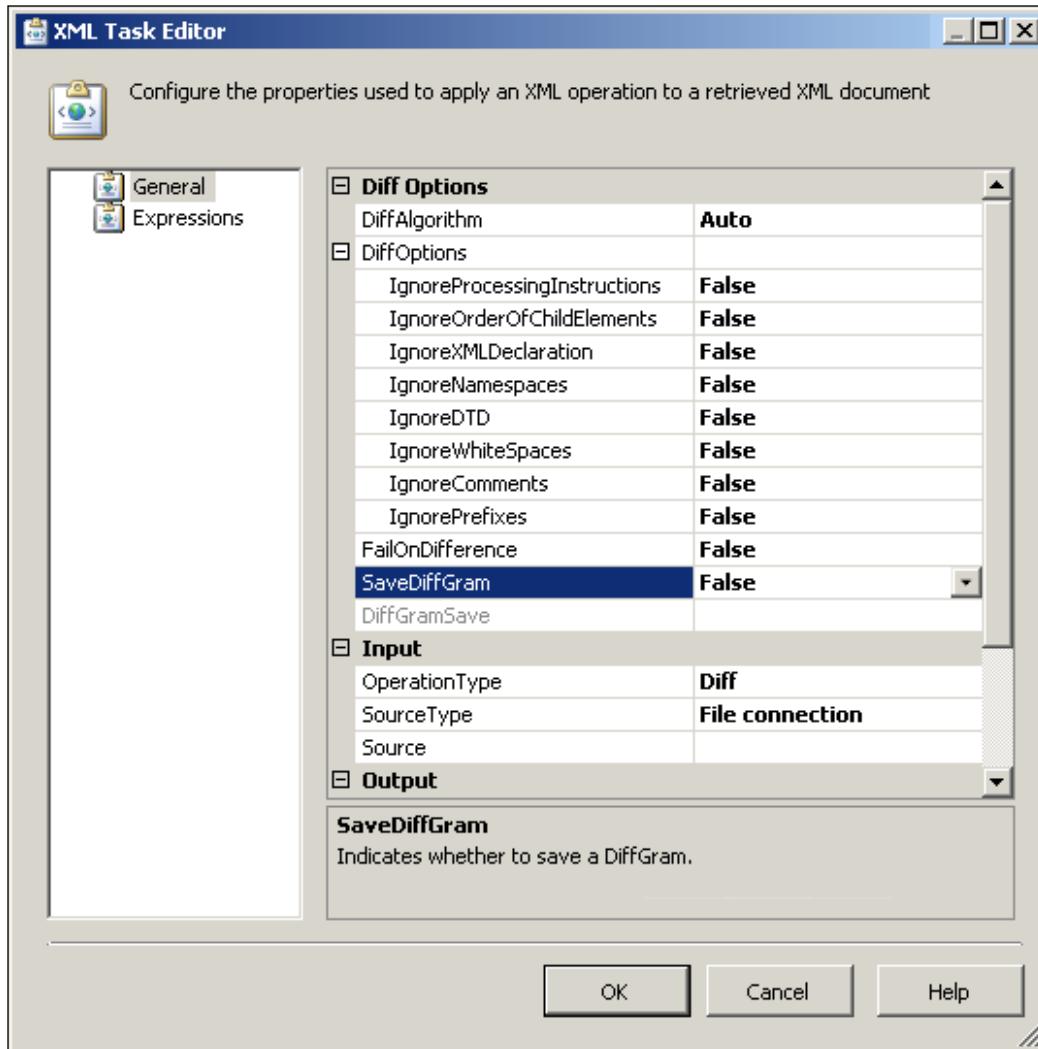
Create a BI project, `Ch11`. Change the name of the default package name to `xmldiff`. `dtsx`. Drag and drop an **XMLTask** from the **Control Flow Items** group in the **Toolbox** to the **Control Flow** page on the **Canvas**. An instance of **XMLTask** will be added to the **Canvas** on the **Control Flow** page.

Step 2: Configuring XMLTask

1. Right-click on the **XMLTask** component on the **Canvas**, and from the drop-down menu, choose **Edit....** This brings up the **XML Task Editor** window, as shown in the following screenshot. It gets displayed with **OperationType**, set to **Diff** as shown.



2. Click on the **DiffOptions** node. This opens the various options that you can choose, as shown in the next screenshot. The default options are all **False**, meaning that none of these are ignored. For example, if for some reason you need to compare two XML documents (ignoring the processing instructions present in the XML documents to be compared), you may do so by setting the first **DiffOptions** to true. Similarly, there are a number of possible options that you can set.



3. In the editor, change the **SaveDiffGram** from **False** to **True**. This will allow saving the **DiffGram** document to a chosen location.
4. Click and expand the **DiffGramSave** node that gets displayed. This will show two more items, **DiffGramDestination** and **DiffGramDestinationType** that need to be specified as shown in the following screenshot. The default for **DiffGramDestination Type** is **File Connection**.

Diff Options	
DiffAlgorithm	Auto
DiffOptions	
FailOnDifference	False
SaveDiffGram	True
DiffGramSave	
DiffGramDestination	
DiffGramDestinationType	File Connection

5. In the **DiffGramSave** node of the previous screenshot, click on **DiffGramDestination**. In the **File Connection Manager Editor** that shows up:
 - Choose to create a file using the **Usage Type: Create File** from the drop-down (the default is **Existing file**).
 - Provide the name **CreatedXMLTsk1.xml** (or any other if you prefer) for the file.
 - Click on the **Browse...** button to open the **Select File** dialog, and find a suitable folder to which you may save the **CreatedXMLTsk1.xml** file.
 - Now click on the **OK** button in the **File Connection Manager Editor**.

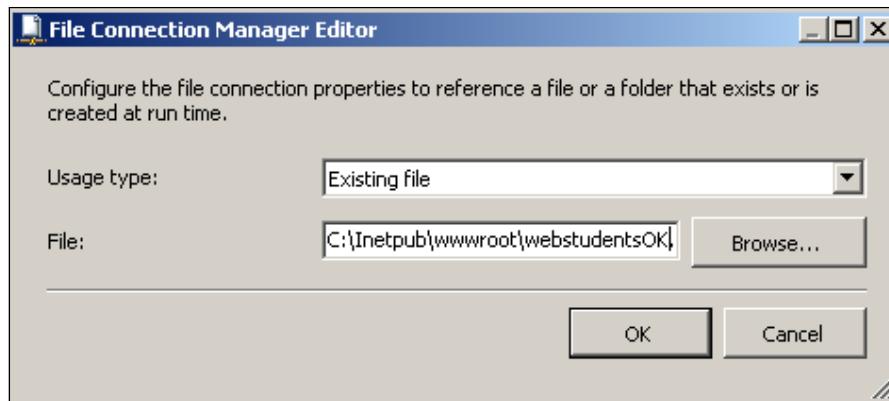
The details of **DiffGramSave** node will be as shown in the following screenshot:

DiffOptions	
FailOnDifference	False
SaveDiffGram	True
DiffGramSave	
DiffGramDestination	CreatedXMLTsk1.xml
DiffGramDestinationType	File Connection

6. Click on the **Input** node to display the line items where you can indicate **SourceType**, as well as **Source**. **SourceType** has three options: **Direct Input**, **File Connection** and **Variable**. We will use **File Connection** option, which also happens to be the default option.

DiffGramSave	
DiffGramDestination	
DiffGramDestinationType	File Connection
Input	
OperationType	Diff
SourceType	File connection
Source	Direct input
Output	
SaveOperationResult	File connection Variable

7. Accept the default source type, and click on an empty area of the **Source** line item, and from the drop-down, click on the item **New File Connection....**. This opens the **File Connection Manager Editor** as shown. In this exercise, you will be using an **Existing file**.
8. Use the **Browse...** button to locate the **webstudentsOK.xml** file on your **C:** drive.
9. Click on the **OK** button after making this choice. The **webstudentsOK.xml** file then appears in the **File Connection Manager Editor** window, as shown in the following screenshot:



10. Click and expand the **Second Operand** node to reveal **SecondOperandType**, and **SecondOperand** items.

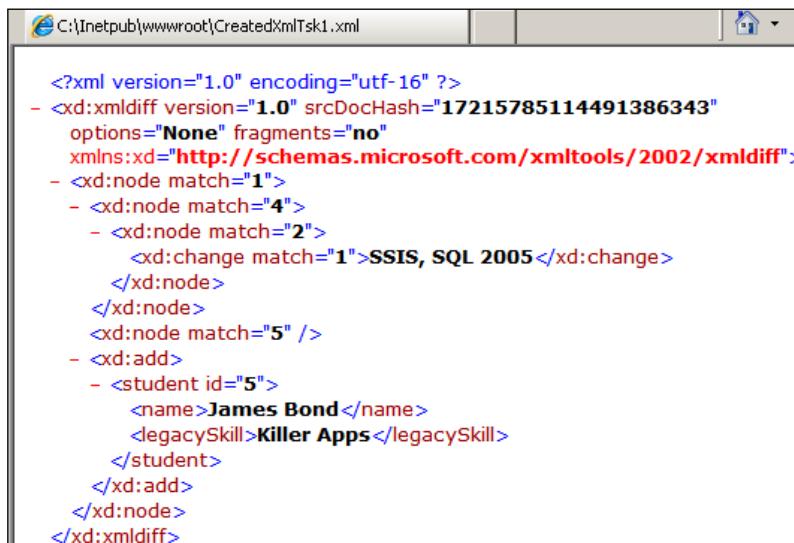
11. Click by the side of **Direct input** for the item, **SecondOperandType**. This will display a list. From there, choose **File Connection**.
12. Click in an empty area in the **SecondOperand** item and choose **New File connection....** This opens the **File Connection Manager Editor**. The editor comes up with default usage type – **Existing file**.
13. Click on the **Browse...** button to open the **Select File** window dialog and choose the second document discussed earlier, and click on the **OK** button. This displays **C:\Inetpub\wwwroot\webstudentsOK2.xml** at **File:** in the **File Connection Manager Editor** window.
14. Click on **OK** in the **File Connection Manager Editor** window. This adds the file name to **SecondOperand** item of the **Second Operand** node. The details of the **SecondOperand** node after these steps should appear as in the following screenshot:

Second Operand	
SecondOperandType	File connection
SecondOperand	webstudentsOK2.xml

Three file connection managers will be added to the **Connection Managers** page and the XML Task Editor appears as shown below. Note that the **Output** node is not configured.

Diff Options	
DiffAlgorithm	Auto
DiffOptions	
FailOnDifference	False
SaveDiffGram	True
DiffGramSave	
DiffGramDestination	CreatedXmlTsk1.xml
DiffGramDestinationType	File Connection
Input	
OperationType	Diff
SourceType	File connection
Source	webstudentsOK.xml
Output	
Second Operand	
SecondOperandType	File connection
SecondOperand	webstudentsOK2.xml

15. Build the project and execute the package. After a few seconds, the XMLTask component turns green on the Canvas. Look for the newly created file, `CreatedXmlTsk1.xml`, in the root directory of your IIS where you saved it. Open this file in the browser. The following XML file is displayed on the browser, showing the differences between the two XML files.



The screenshot shows a Microsoft Internet Explorer window with the URL `C:\Inetpub\wwwroot\CreatedXmlTsk1.xml`. The page content displays an XML document representing the differences between two XML files. The XML code includes various nodes and changes, such as additions, deletions, and modifications, with specific node IDs and match numbers highlighted in red and blue.

```
<?xml version="1.0" encoding="utf-16" ?>
- <xd:xmldiff version="1.0" srcDocHash="17215785114491386343"
  options="None" fragments="no"
  xmlns:xd="http://schemas.microsoft.com/xmltools/2002/xmldiff">
- <xd:node match="1">
- <xd:node match="4">
- <xd:node match="2">
  <xd:change match="1">SSIS, SQL 2005</xd:change>
</xd:node>
<xd:node match="5" />
- <xd:add>
- <student id="5">
  <name>James Bond</name>
  <legacySkill>Killer Apps</legacySkill>
</student>
</xd:add>
</xd:node>
</xd:xmldiff>
```

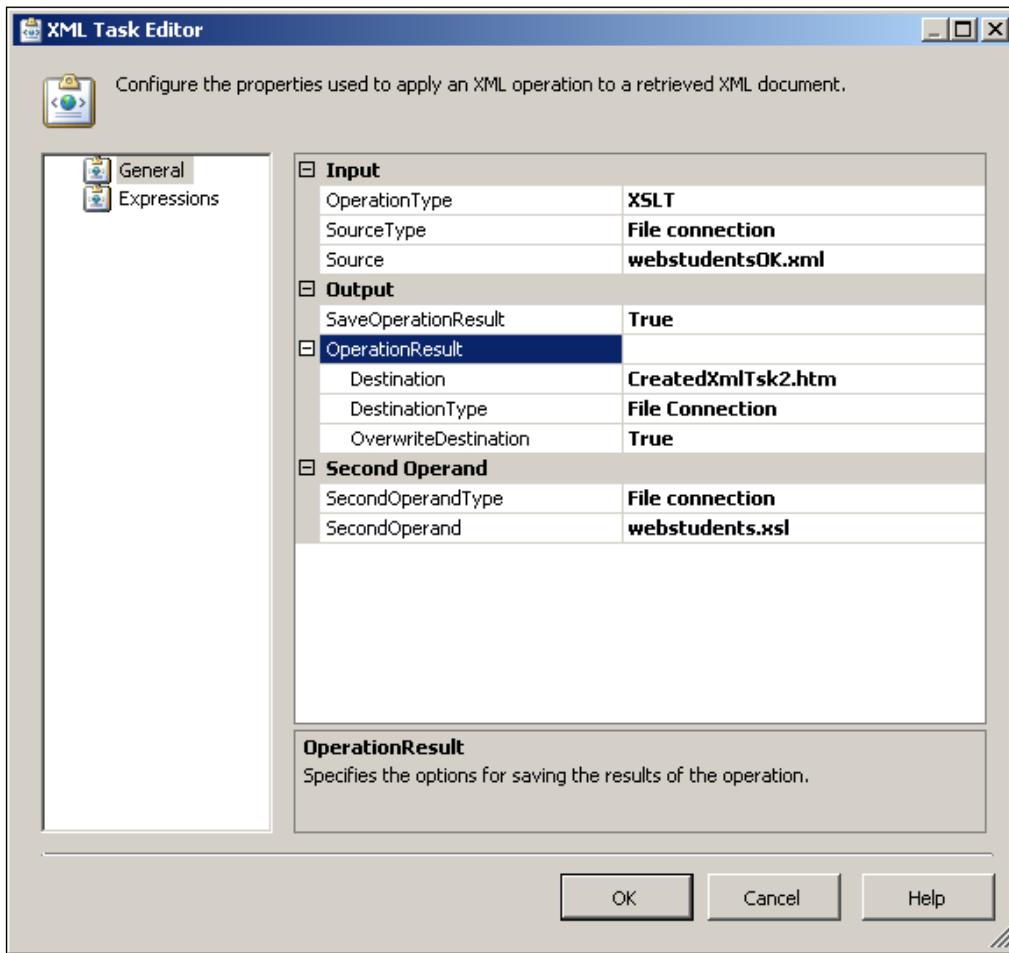
Hands-On Exercise 2: XMLTask Type XSLT

In this exercise we will transform the `webstudentsOK.xml` to an HTML format displayed by a `CreatedXmlTsk2.htm` page using a transformation file, `webstudents.xsl` described earlier.

The configuration follows the same steps as you need to access the **File Connection Manager Editor**, each time you want to use an existing file, or create a new file on your file system. In this exercise, you will create the `CreatedXmlTsk2.htm` file in the root directory of your IIS (saving it to IIS is not essential and it can be stored in any directory of your choice). The completed **XML Task Editor** should appear as shown here.

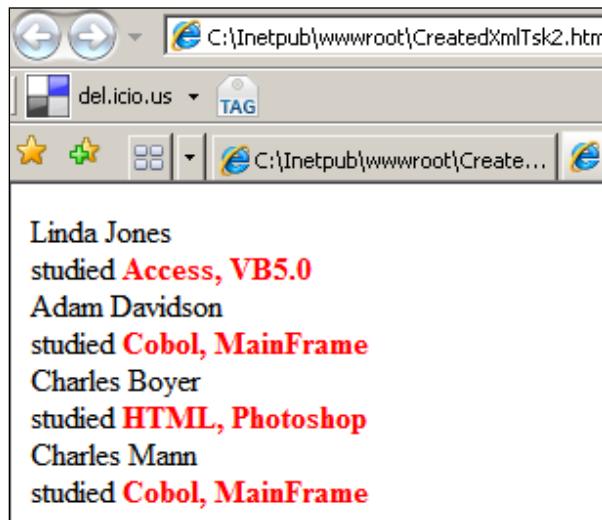
16. Right-click on **XMLTask** of **Hands-on Exercise 1**, and from the drop-down menu, disable it. Add a new **XMLTask** from the **Toolbox** on to the **Control Flow** page in the Canvas. You can disable a task which makes it inactive. This was disabled in this exercise in lieu of creating another BI project all over.

17. Edit the **XML Task Editor** so that it looks the same as the one shown in the next screenshot. For the **Destination** line item in the node **OperationResult**, you would choose to create a file (not use an existing one) and provide a name and location.



18. Build the project and execute the XMLTask created for the **OperationType**, **XSLT**. The project runs with this new XMLTask and afterwards turns green indicating success.

19. Now bring up the file C:\Inetpub\wwwroot\CreatedXmlTsk2.htm in your browser (or from your chosen saved location). This is displayed as shown in the following figure:



Summary

This chapter described in detail, the steps involved in using XMLTask that compares two XML documents and writing the difference to another XML document. The XML Task with the operation type XSLT that applies XSL transformation to an XML file was also described. This would have normally required you to code in Javascript or other scripting or programming language to access the properties and methods of the DOM [Document Object Model]. Visit W3C (<http://www.w3c.org/xml/>) to learn about XML, and how to code an XSL file.

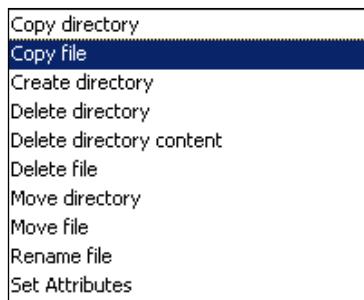
12

Creating a SSIS Package to Access Folders and Files

Access to host computer's folder/file system is an important desired feature in any programming language. Visual Basic provided this functionality to access folders and files (that means virtual directories on the server) with the **FileSystemObject** object. In Visual Basic, you would have used the `CreateObject ("Scripting.FileSystemObject")` to create this object. Once created, you would then use all its properties and methods.

There is yet another method, the early binding method, that declaratively creates this object by referencing the Microsoft Scripting Runtime library (`scrrun.dll`) to instantiate a new **Scripting.FileSystemObject**. If you would like to understand the **FileSystemObject** take a look at this http://www.w3schools.com/asp/asp_ref_filesystem.asp tutorial. Although access to folder and files are simplified by the **FileSystemObject**, caution must be exercised as viruses can enter when the file/folder structure of the computer are given easy access.

The File System Task in SSIS can do a subset of the operations which were called "methods" in Visual Basic's **FileSystemObject** (<http://msdn2.microsoft.com/en-us/library/z9ty6h50.aspx>). The following screenshot shows the type of operations that the File System Task can carry out in SSIS.



In this chapter, in addition to using the File System Task, you will also learn about Precedence Constraints, a programming construct necessary for ordering tasks in a package. When there are a number of tasks, for the flow from one task to the other to be smooth and free of problems, some ordering is needed. Like you have to apply for a job before you get interviewed. You get hired only after the interview, etc. The Precedence constraint in SSIS does the same with executables (For Loop, Task, Event Handlers, etc.), containers, and tasks in packages.

Hands-On Exercise: Copying a File from One Folder to Another and Sending it using the Send Mail Task

In this exercise, we will use the File System Task to copy an ASCII file from one folder to another folder using the **CopyFile** operation. When this task succeeds, we will email this using the **Send Mail Task**, a task we studied in an earlier chapter.

In completing this exercise, you will be carrying out these major steps:

- Creating a BI Project and adding a Control Flow Task – the File System Task.
- Configuring the File System Task.
- Adding a Send Mail Task to the Control Flow page on the Canvas.
- Adding a Precedence Constraint.
- Building and Executing the package.

Step 1: Creating a BI Project and Adding a Control Flow Task — the File System Task

This process has been described a number of times in the previous chapters.

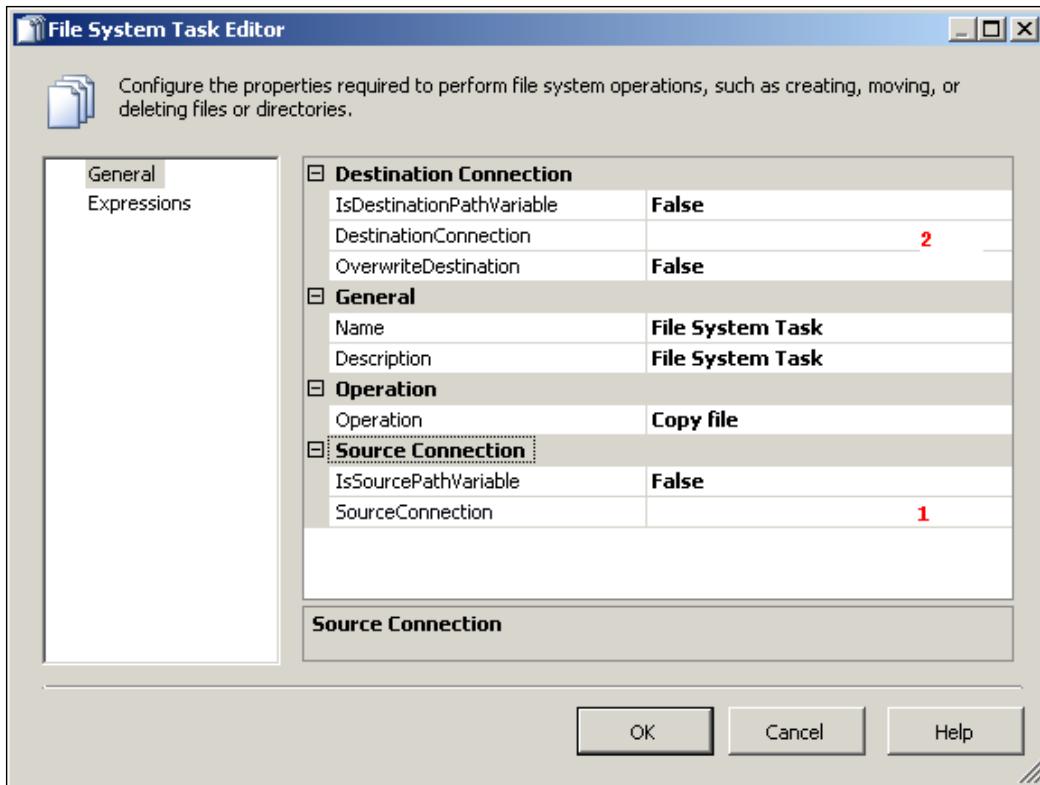
1. Create a BI project **Ch 12**. Change the name of the default package name to **FileSystemTsk.dtsx**. Drag and drop a **File System Task** from the **Control Flow Items** group in the **Toolbox** to the **Control Flow** page on the **Canvas**.

An instance of the **File System Task** will be added to the **Canvas** to the **Control Flow** page.

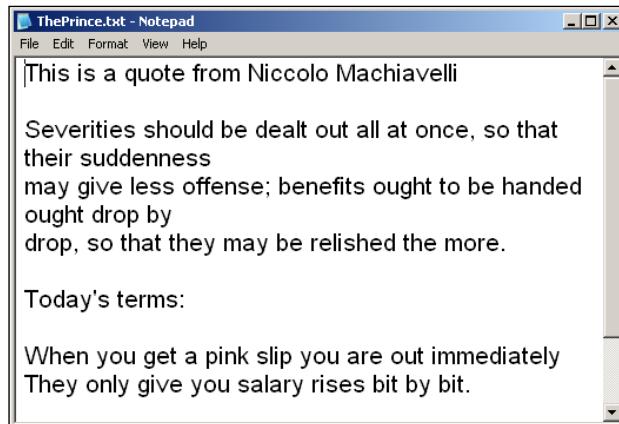
Step 2: Configuring The File System Task

1. Right-click the **File System Task** component on the Canvas and from the drop-down menu choose **Edit....**

This brings up the **File System Task Editor** window as shown in the next screenshot. In the previous screenshot, we saw several operations possible with this task. **CopyFile** operation appears by default when this editor opens.



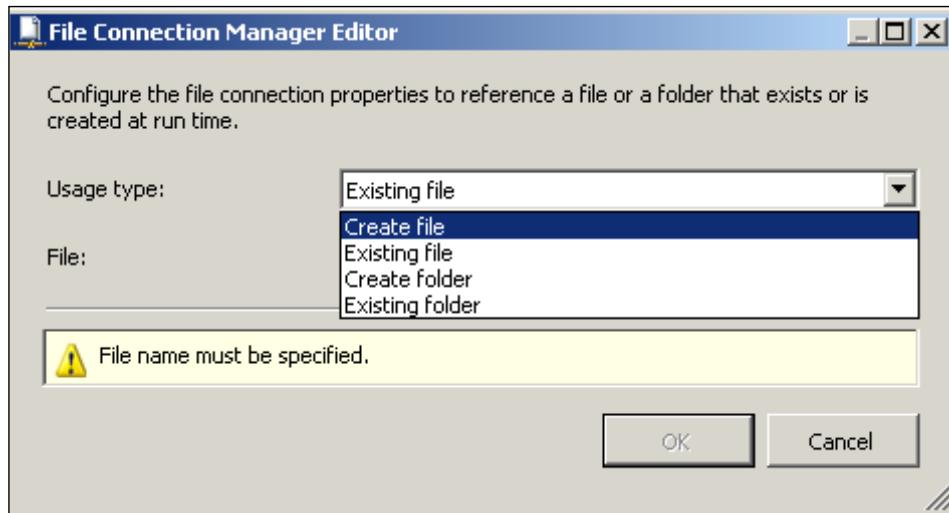
We need to indicate the source file and its location as well as the destination location to which this file will be copied. File System Task would interact with the file system through connection managers that you can designate at design time, as we saw in Chapter 11. We use exactly the same procedure to locate the file to be copied. The file we are going to use is *ThePrince.txt* located on the C:\ drive, a copy of which is shown in the following screenshot. The reader may use his/her own text file instead of the previous file.



2. Click on a point shown by **1** in the line item **SourceConnection**.
A drop-down handle develops.
3. Click on the drop-down and choose <**New connection...>**
This opens the **File Connection Manager Editor** with the default **Usage Type:** as **Existing file**.
4. Click on the **Browse...**button.
This opens the **Select File** dialog window where you can browse through folders and files.
5. Locate the file C:\ThePrince.txt and click open.
The file name appears in the **File:** line item left of the **Browse...** button.
6. Click on the **OK** button in the File Connection Manager Editor's window.
This will display the file name in the line item, **SourceConnection** in the same location as the 1 shown in the screenshot. A connection manager, *ThePrince.txt*, gets added to the **Connection Managers**' page on the Canvas.

7. Click on the point shown by **2** in the line item **Destination Connection**.

In the **File Connection Manager Editor** that opens, instead of **Existing file**, choose **CreateFile** option, as shown in the next screenshot.



8. Click on the **Browse...** button and create a file `CopiedPrince.txt` in the **My Documents** folder. Then click on the **OK** button.

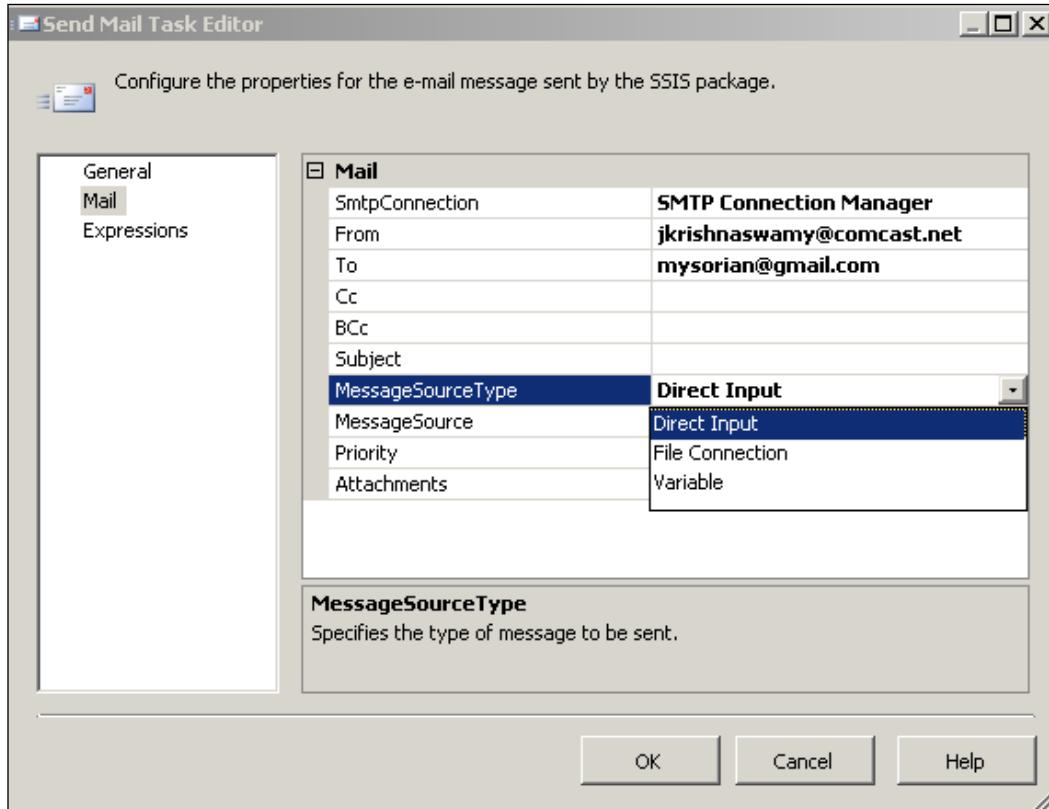
This displays the `CopiedPrince.txt` to the **Destination Connection** line item at position shown by **2** in the screenshot. A `CopiedPrince.txt` connection manager gets added to the **Connection Managers**' page on the Canvas.

9. Click on the **OK** button in the File System Task Editor's window to complete this task.

At this point, you should be able to Execute the Package as we have done earlier, or execute the File System Task (by right-clicking this control on the Canvas and choose **Execute Task**). Once this task has completed, you should be able to locate the `CopiedPrince.txt` file in the **My Documents** directory (perhaps you may have to refresh the view).

Step 3: Adding a Send Mail Task to the Control Flow Page on the Canvas

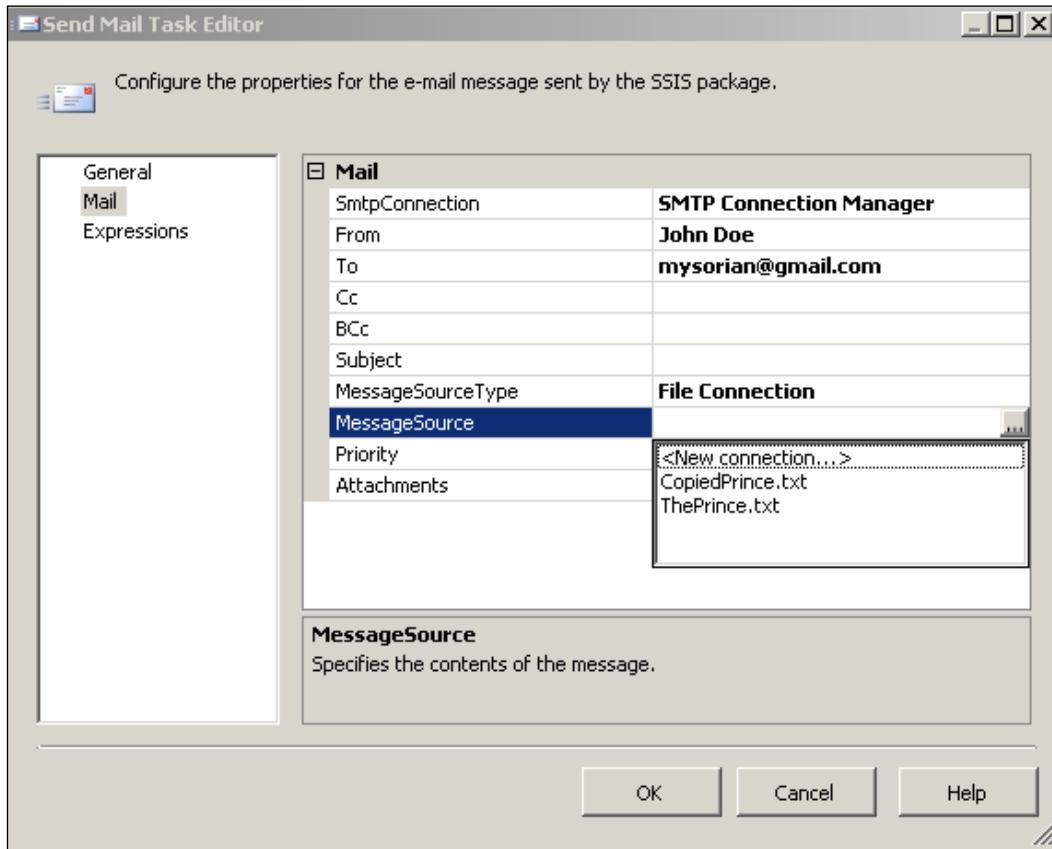
Please read up Chapter 3 where the Send Mail Task was discussed. The next screenshot shows a partially configured Send Mail Task that will email a file to mysorian@gmail.com.



Since **File Connection** option will be chosen for the **MessageSourceType**, a **MessageSource**.

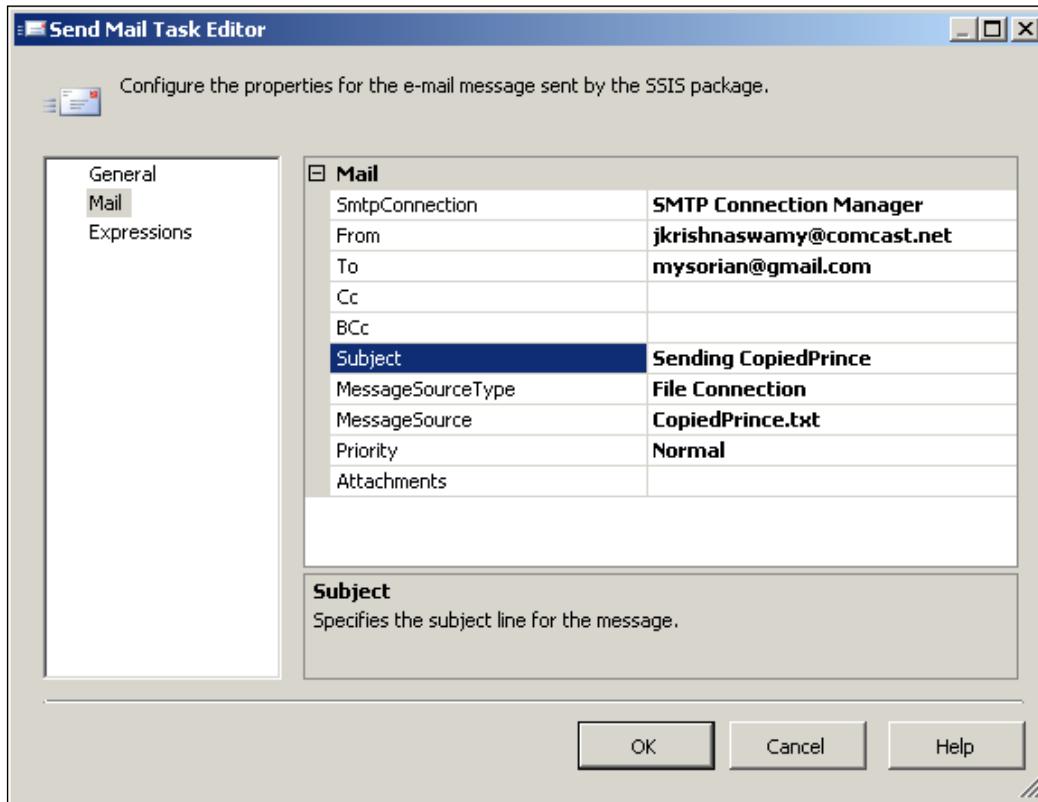
1. Click on an empty area along the line item **MessageSource** (in the second column), which enables a drop-down handle. Click on this handle.

The drop-down menu shows the existing two file system connection managers, as well as an option to create a new connection.



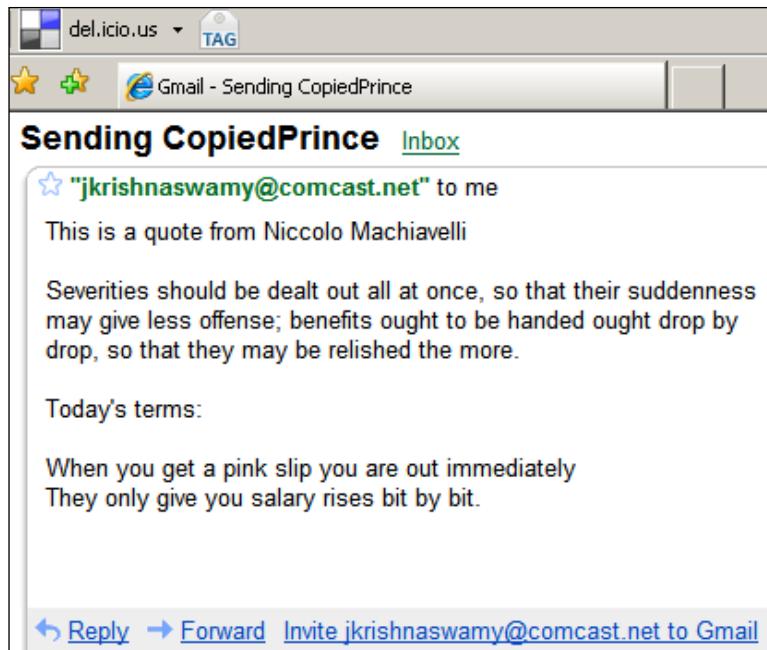
2. Click on CopiedPrince.txt since this was the one copied by the File System Task. Now for the line item **Subject** type in some text (Sending CopiedPrince in here).

The completed Send Mail Task is as shown below.



3. Click on the **OK** button in the **Send Mail Task Editor** and you are done.

Now you can execute the Send Mail Task, which will email the file shown in the above editor to the destination email address.



Using Precedence Constraint to Send Mail

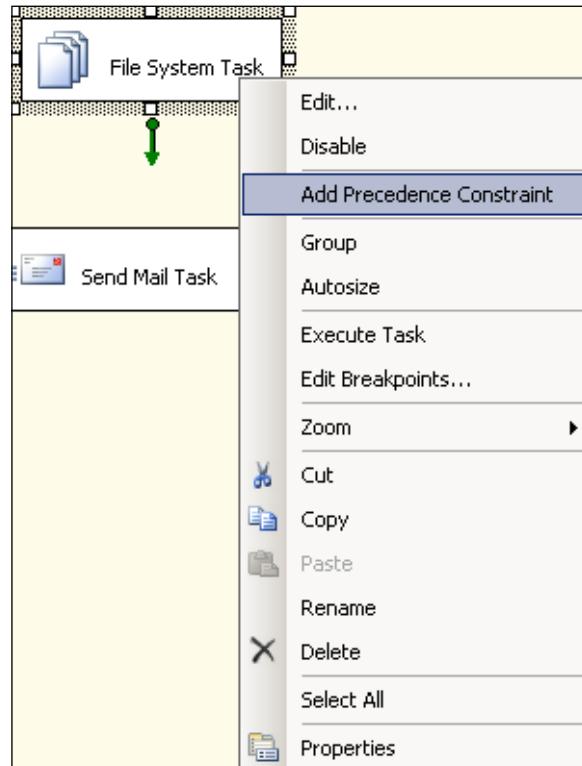
In the present exercise, the **File System Task** and the **Send Mail Task** are two executables (you can independently execute) and we want the File System Task to create the file in the chosen destination and only then, the Send Mail Task will email this file. This kind of precedence will depend on execution being a success, failure, or task completion.

The line item in the **Destination Connection** node for **OverwriteDestination** is set as **False**. This means if the task were to be executed a second time the task would fail because the file exists and it cannot be overwritten.

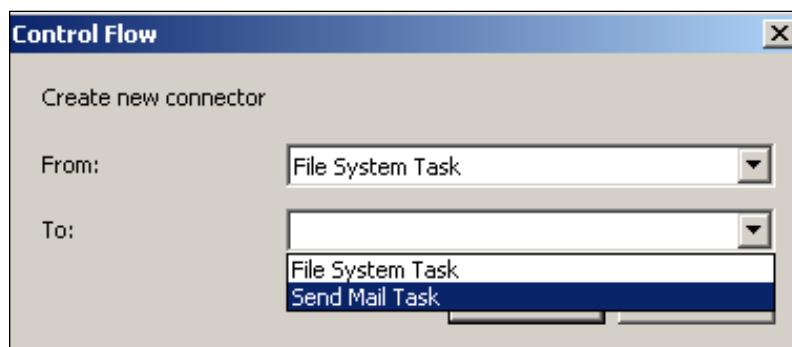
1. Go to Windows Explorer and delete `CopiedPrince.txt`.

Step 4: Adding a Precedence Constraint

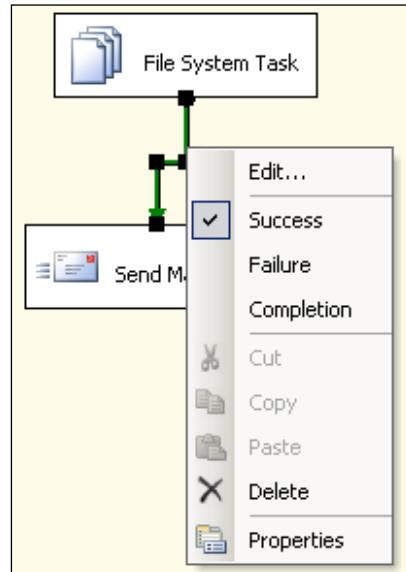
1. Right-click the **File System Task** on the Canvas and from the drop-down choose **Add Precedence Constraint**, as shown in the next screenshot.



This opens the **Control Flow** window showing **From:** and **To:** windows with the **From:** already chosen as **File System Task**. The drop-down for **To:** reveals both the **File System Task** as well as the **Send Mail Task**.



2. Choose the **Send Mail Task** and click on the **OK** button.
A green line now connects the **File System Task** and the **Send Mail Task**
3. Right click on this green line as shown in the following screenshot.



The default execution constraint is Success which means that the Send Mail will carry out its task after the File System Task has successfully completed its task.

Step 5: Building and Executing the Package

This has been described several times in the earlier chapters.

1. Right-click the **FileSystemTsk.dtsx** in the **Solution Explorer** and execute the package.

First the **File System Task** turns green while the **Send Mail Task** will change from yellow to green signifying that the package succeeded. You can verify that the mail has been sent.

Summary

This chapter described in detail the steps involved in using a File System Task to copy a file from one directory to another directory. The steps need to send this copied file by email was described, and in doing so the concept of Precedence Constraint and its usage was described.

13

Package to Copy a Table from Oracle XE

Modern business enterprises have to contend with resources from disparate vendors of databases for many different reasons, that includes historical: new developments in technology; cost concerns; acquisition of resources (M & R), etc. For example, although a large hospital might have its main database using Oracle or SQL Server 2005, the various departments and cost centers may use other products. In these cases, it is important to have the capability of easily transferring objects from one vendor type product to the other. Every database vendor provides a built-in toolset to address the migration of database objects, or whole databases to another vendor type database.

Although this tutorial describes the details of copying a *Table* from Oracle 10G XE to an SQL Server 2005 database using a SSIS task, you should review an earlier article (<http://www.devshed.com/showblog/28087/StepByStep-Guide-to-Importing-Data-from-Oracle-XE-to-SQL-2005/>), which uses the IMPORT / EXPORT utility to import data from Oracle 10G XE to SQL Server 2005.

In this exercise, we will transfer a Table from the **hr** database on the local Oracle 10G XE server to the *MyNorthwind* database on the local SQL Server 2005 using a Data Flow Task. In order to follow the steps indicated, you will need a source and a destination—the source, data extracted from an Oracle 10G XE server and the destination, loading this to *MyNorthwind* database on the SQL Server 2005. You also need to establish a path connecting them.

Hands-On Exercise: Transferring a View from Oracle 10G XE to an SQL Server 2005 Database

The following are the major steps in this exercise:

- Installing and viewing objects on the Oracle 10G XE.
- Creating a BI project and adding a Data Flow Task.
- Adding an OLE DB Source and configuring it to connect to a local Oracle 10G XE Server.
- Configuring the OLE DB Source.
- Adding an SQL Server Destination and configuring its connection manager.
- Establishing a path from the OLE DB Source to the SQL Server Destination.
- Configuring the SQL Server Destination Component.
- Building and executing the package.

Oracle 10G XE Server

The Oracle 10G XE is available as a free download from Oracle's website (<http://www.oracle.com/technology/software/products/database/xe/index.html>). Downloading and installing Oracle 10G XE is fully described in the tutorial (<http://www.devshed.com/c/a/Oracle/Experience-the-Possibilities-with-Oracle-10g-Express-Edition>).

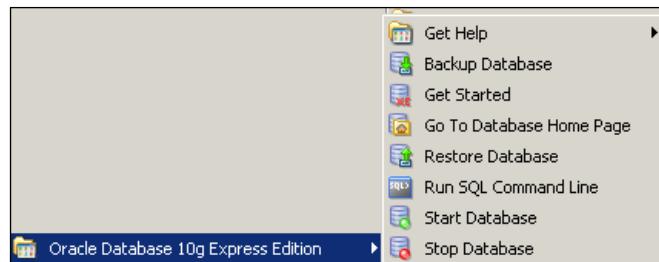
Starting and Stopping the Oracle 10G XE Server

With the Oracle 10G XE installed, you should be able to configure its stop / start mode using the Window's Services. From **Start** you can get to the **Services** folder after clicking **Control Panel**, **Administrative Tools**, and **Services** in succession. This opens the **Services** window as shown in the next screenshot where you can scroll down to the Oracle XE Server. Here, you can click on the hyperlinks in the left to stop, restart or pause the service.

The screenshot shows the Windows Services (Local) window. The OracleServiceXE service is listed with the following details:

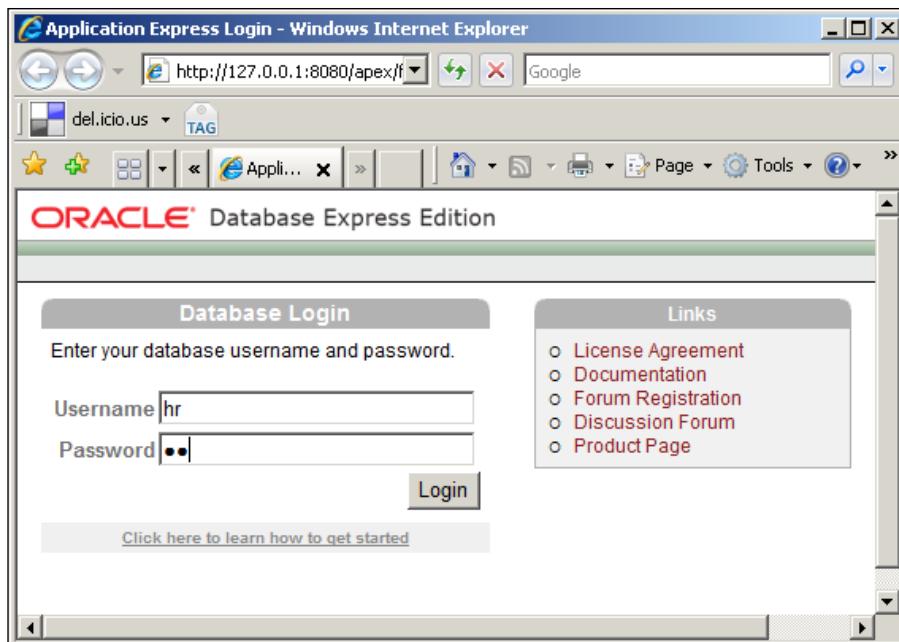
Name	Description	Status	Startup Type	Log On As
Office Source Engine	Saves inst...	Manual	Local System	
OracleJobSchedulerXE		Manual	Local System	
OracleMTSRecover...		Manual	Local System	
OracleServiceXE		Started	Automatic	Local System
OracleXEClrAgent		Manual	Local System	
OracleXETNSListener		Started	Automatic	Local System
Performance Logs a...	Collects pe...	Manual	Network S...	

Another convenient way is to use the shortcuts installed when the Oracle 10G XE is installed, as shown in the next screenshot. Some times the short-cuts could get displaced and it is a better practice to look up in the **Services** folder. The shortcut also gives you access to the home page of the database.



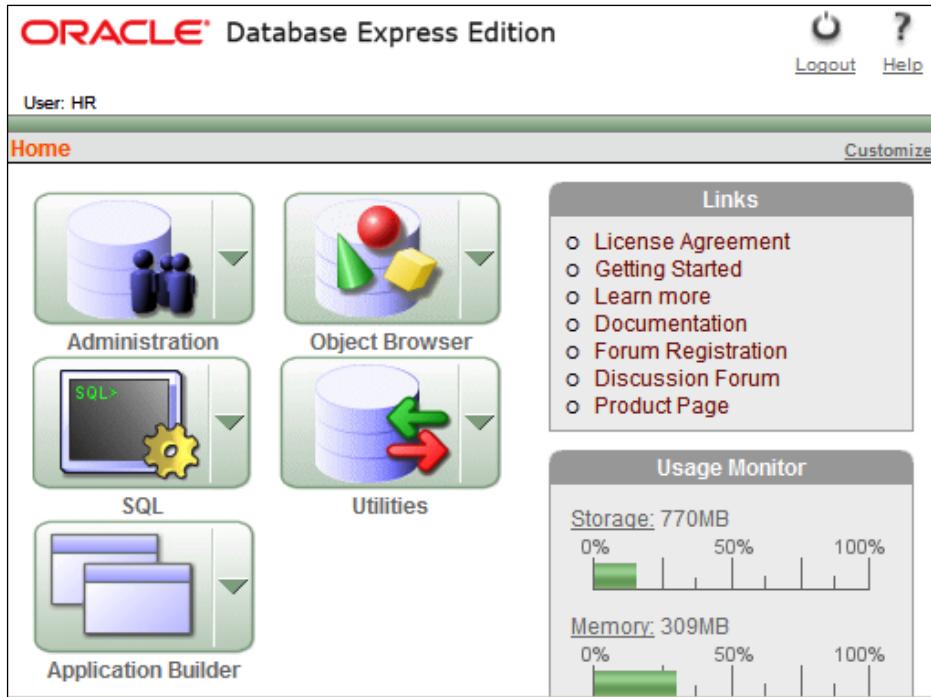
Using the Object Browser

Go To Database Home Page shortcut takes you to the Oracle 10G XE database. Using this HTML page, you will be able to administer, as well as use, its various tools discussed in the link provided at the beginning of this section. You can find links to a number of tutorials all dealing with this database at the author's blog (<http://hodentek.blogspot.com/2006/11/links-to-my-oracle-10g-xe-articles.html>). When the browser opens up, as shown in the next screenshot, you need to provide the username and password, which are **hr** and **hr**.



Package to Copy a Table from Oracle XE

When you click on the button **Login**, you will open the **Home** page as shown in the next screenshot. You may click on the **Object Browser** component in the **Home** page.



The **Object Browser** is the user interface to do everything with the database objects. Here, you can review tables, views, stored procedures, etc. In the drop-down menu with the **Tables** chosen, you will be able to see the details of the Table, **EMPLOYEES**, as shown in the next screenshot.

The screenshot shows the Oracle Database Express Edition Object Browser interface. The left sidebar lists various database objects under the 'Tables' category, including ARGENTINA, COUNTRIES, DEPARTMENTS, EMPLOYEES (which is selected and highlighted in green), EM_DE_L, EmployeesCSV, FIRST_LAST, HTEK_EMPLOYEES, HTMLDB_PLAN_TABLE, JOBS, JOB_HISTORY, LOCATIONS, NUMB, NW3_Emps, NW4_EMPS, and NorthEmployees. The main right panel displays the structure of the EMPLOYEES table. The table has 10 columns: EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE, JOB_ID, SALARY, COMMISSION_PCT, and MANAGER_ID. The PHONE_NUMBER column is defined as VARCHAR2(20) and is nullable (Yes). The SALARY column is defined as NUMBER(8,2) and is nullable (Yes). The COMMISSION_PCT column is defined as NUMBER(2,2) and is nullable (Yes). The MANAGER_ID column is defined as NUMBER(6,0) and is nullable (Yes). The DEPARTMENT_ID column is defined as NUMBER(4,0) and is nullable (Yes).

Column Name	Data Type	Nullable	Default
EMPLOYEE_ID	NUMBER(6,0)	No	-
FIRST_NAME	VARCHAR2(20)	Yes	-
LAST_NAME	VARCHAR2(25)	No	-
EMAIL	VARCHAR2(25)	No	-
PHONE_NUMBER	VARCHAR2(20)	Yes	-
HIRE_DATE	DATE	No	-
JOB_ID	VARCHAR2(10)	No	-
SALARY	NUMBER(8,2)	Yes	-
COMMISSION_PCT	NUMBER(2,2)	Yes	-
MANAGER_ID	NUMBER(6,0)	Yes	-
DEPARTMENT_ID	NUMBER(4,0)	Yes	-

Step 1: Creating a BI Project and Adding a Data Flow Task

This process has been described a number of times in the previous chapters.

1. Create a BI project Ch13. Change the name of the default package name to OraTo2k5.dtssx. Drag and drop a **Data Flow Task** from the **Control Flow Items** group in the **Toolbox** to the **Control Flow** page on the Canvas.

An instance of the **Data Flow Task** will be added to the Canvas to the **Control Flow** page.

Step 2: Adding an OLE DB Source and Configuring it to Connect to a Local Oracle 10G XE Server

1. Drag and drop a **OLE DB Source** from the **Data Flow Sources** group in the **Toolbox** to the **Data Flow** page of the Canvas.
2. Right-click an empty area in the **Connection Managers**' page in the Canvas, and from the pop-up menu choose **New OLE DB Connection...**

(The detailing of what to do next is abbreviated as this has been dealt in an earlier chapter). This opens the **Configure OLE DB Connection Manager**'s window. The left-hand area shows all existing data connections under **Data Connections:** label and the right-hand area shows the details of the data connection you choose on the left under the label **Data connection properties**. On the right side, below you will find the **New...** button to create a new OLE DB Connection.

3. Click on the **New...** button.

This opens the **Connection Manager**'s window that opens as a default connection to the SQL Server 2005 using the SQL Native Client provider, and as such this window has controls suitable to connect to an SQL Server. However, to connect to an Oracle database we need to use a **Provider** for Oracle, which is chosen using the drop-down menu item corresponding to the **Provider** label on this page.

4. Click on the drop-down menu and choose the Microsoft OLE DB Provider for Oracle, as shown in the next screenshot.

The next figure shows the **Connection Manager**'s page as well as the result of clicking the **Test Connection** button in the mini-step 5 to follow.

5. Enter **Xe** for the **Server name**, **hr** for **username**, and **hr** for **password**. Also place a check mark for the **Save my password** check box.. Click on the **Test Connection** button.

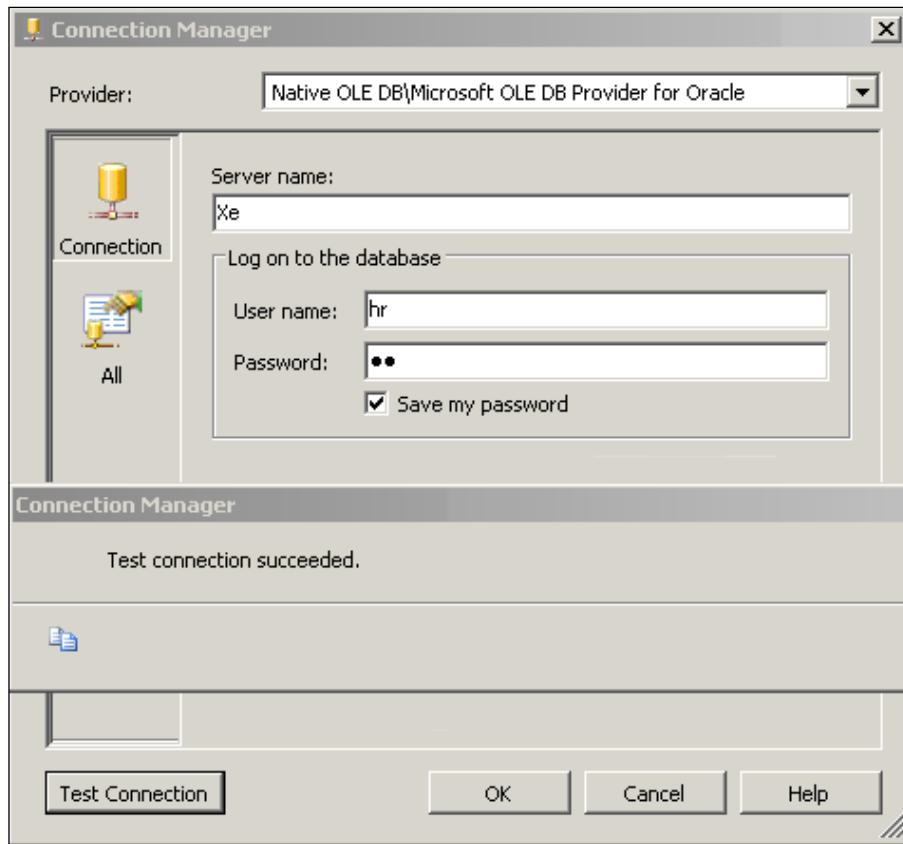
As the information supplied is correct, the connection to the Oracle 10G XE was successful.

6. Click on the **OK** button on the **Test Connection** button generated message as well as the **Connection Manager** window.

This will bring you back to the **Configure OLE DB Connection Manager**'s window. A new OLE DB Connection **xe.hr** gets added to the Data Connections area in the left.

7. Click on the **OK** button on this window.

This completes the connection manager setup for the Oracle 10G XE. A connection manager **xe.hr** appears in the **Connection Managers**' page on the Canvas.



Step 3: Configuring the OLE DB Source

1. Right-click the **OLE DB Source** component on the canvas and bring up the **OLE DB Source Editor**, as shown in the next screenshot.

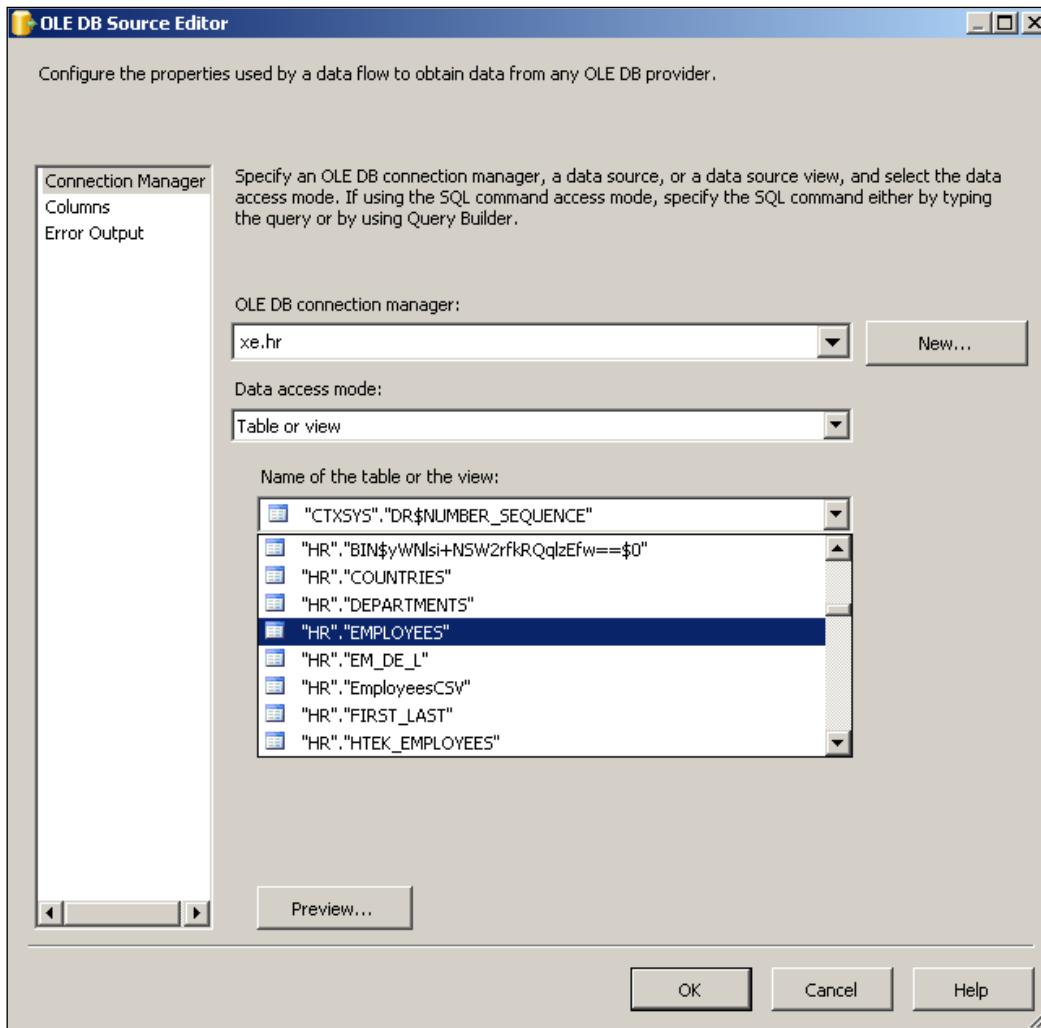
The OLE DB Connection Manager is the recently created connection manager **xe.hr**. As we are accessing a *Table* from the database, the default for the **Data access mode:** is acceptable as it is.

2. Click on the drop-down for the **Name of the table or view:**

This field reveals all the accessible objects on the hr database on the Oracle 10G XE Server.

Package to Copy a Table from Oracle XE

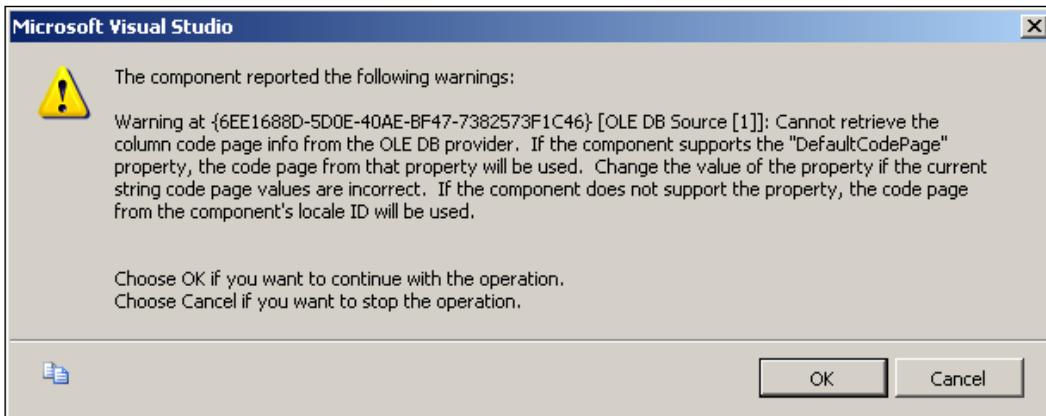
3. Choose the view, "HR"."EMPLOYEES", we saw earlier in the object browser of this database.



4. Click on the **Preview...** button to see the *Table* that is being used from the Oracle 10G XE through this connection.

This brings up a **Microsoft Visual Studio** warning page as shown in the following screenshot. Ignoring this warning and continuing by clicking on the **OK** button on the window with the warning will show all the columns of the EMPLOYEES in the **Preview Query Results** window.

5. Click the **OK** button.

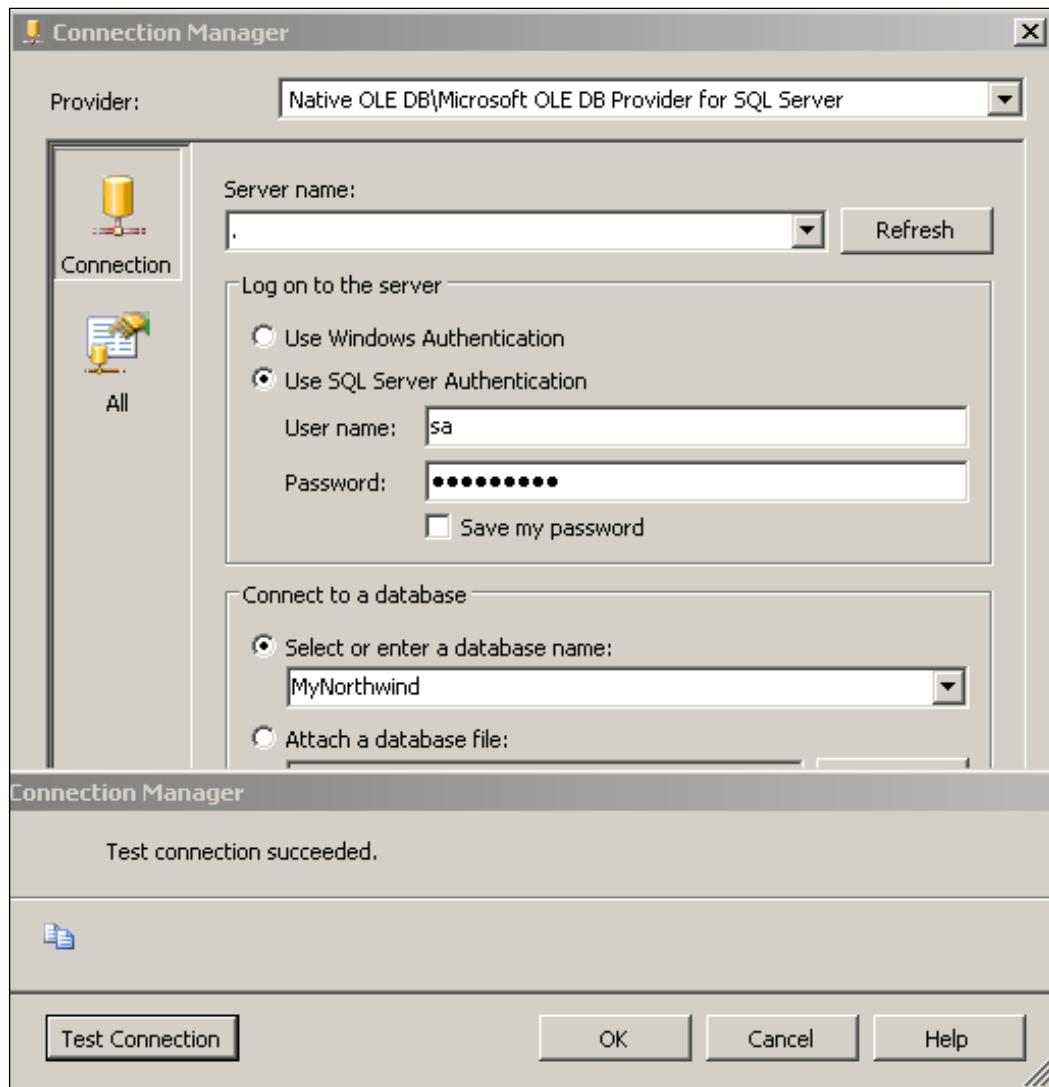


Step 4: Adding a SQL Server Destination and Configuring its Connection Manager

1. Drag and drop an **SQL Server Destination** component from the **Data Flow Destinations** group, from the **Toolbox** onto the **Data Flow** page of the Canvas.
2. Following a procedure similar to the OLE DB Connection Manager, configure a Connection Manager (localhost.MyNorthwind) for the **SQL Server Destination**.

Package to Copy a Table from Oracle XE

The following screenshot shows the final screen of the Connection Manager window, after the connectivity test using the **Test Connection** button. The **Provider** chosen is the Microsoft OLE DB Provider for SQL Server. You may notice that the (.) format is used for designating a default SQL Server 2005 installation. This server uses SQL Server Authentication. After choosing the authentication, you will be able to **Select or enter database name** from a drop-down list. In case your server uses Windows authentication, you don't need to provide the username and password.

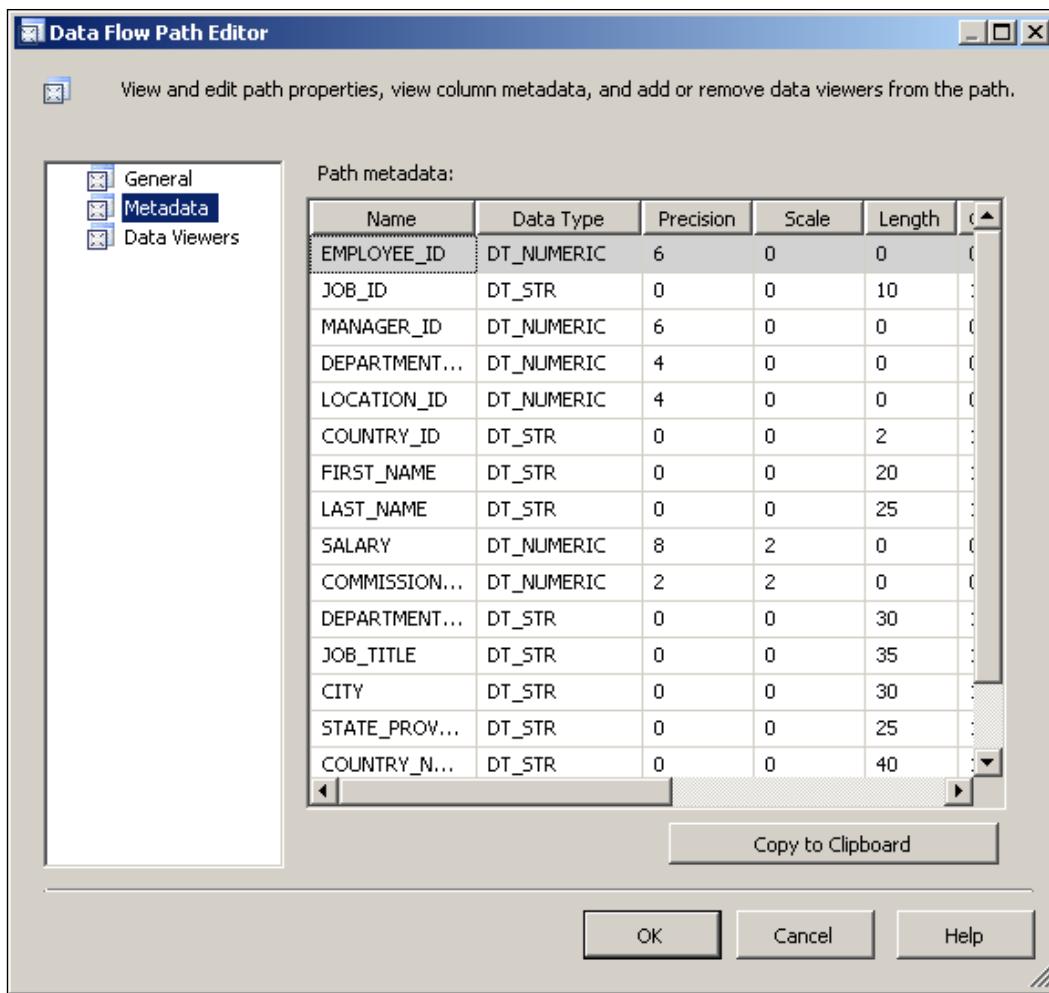


Step 5: Establishing a Path from the OLE DB Source to the SQL Server Destination

This has been described in most of the tutorials , so wont be repeated here.

1. Establish a path from the **OLE DB Source** to the **SQL Server Destination**.

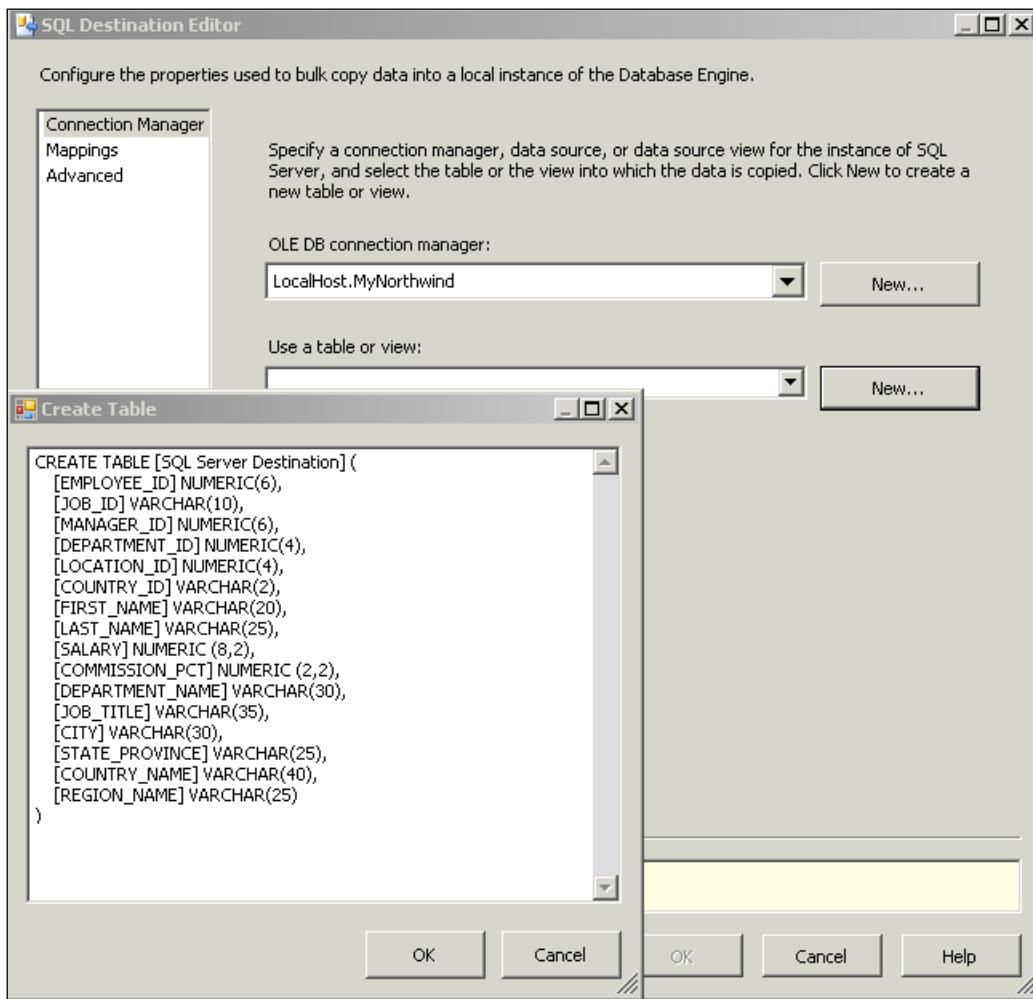
The **Metadata** of the path connecting the source and the destination will appear as shown in the **Data Flow Path Editor** window below.



Step 6: Configuring the SQL Server Destination Component

1. Right-click the **Sql Server Destination** component and from the drop-down click on **Edit....**

This opens up the **SQL Destination Editor** displaying the connection manager configured earlier.

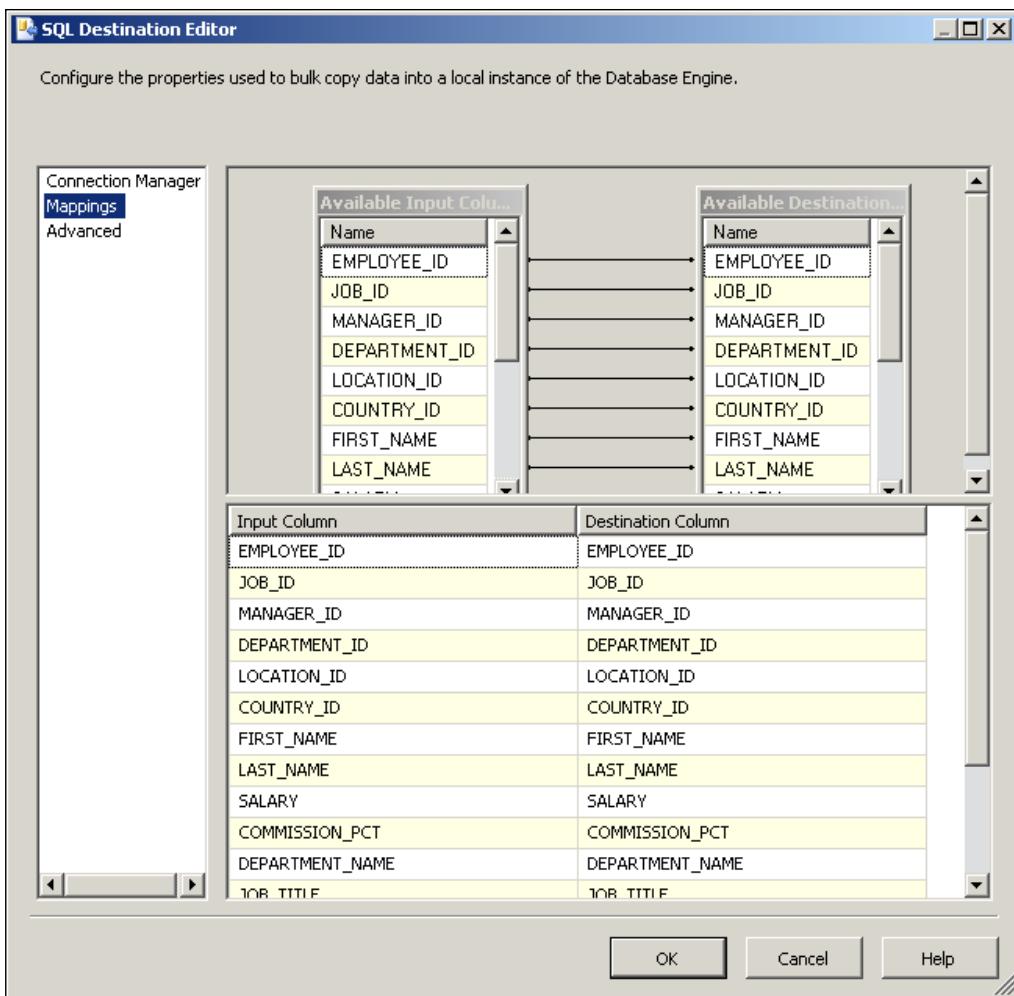


2. Click on the **New...** button. "Use a table or view:".

This pops-up the **Create Table** window shown superposed on the **SQL Destination Editor**.

3. In the **Created Table** window, replace the **SQL Server Destination** by some name of your own. Here, for this exercise, it is called **Oracle Table**. Click on the **OK** button on the **CreateTable** window after changing SQL Server Destination to OracleTable.
4. Now that the connection manager and the table are chosen, click on the **Mappings** list item on the left.

This opens the Mappings between the **View** on the Oracle 10G XE and the **Available Destination Columns**.



5. Accept this default mapping and click on the button **OK** on the above editor to complete the **SQL Server Destination** configuration.

Step 7: Building and Executing the Package

1. Build and execute the package.

The program runs and after a while the source and destination components turn green indicating that it was a successful run. Open the **OracleTable** table in the SQL Server to verify that data has been transferred as shown below.

Table - dbo.OracleTable								
	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
▶	100	Steven	King	SKING	515.123.4567	6/17/1987 ...	AD_PRES	24000.00
	101	Neena	Kochhar	NKOCHHAR	515.123.4568	9/21/1989 ...	AD_VP	17000.00
	102	Lex	De Haan	LDEHAAN	515.123.4569	1/13/1993 ...	AD_VP	17000.00
	103	Alexander	Hunold	AHUNOLD	590.423.4567	1/3/1990 1...	IT_PROG	9000.00
	104	Bruce	Ernst	BERNST	590.423.4568	5/21/1991 ...	IT_PROG	6000.00
	105	David	Austin	DAUSTIN	590.423.4569	6/25/1997 ...	IT_PROG	4800.00
	106	Valli	Pataballa	VPATABAL	590.423.4560	2/5/1998 1...	IT_PROG	4800.00
	107	Diana	Lorentz	DLORENTZ	590.423.5567	2/7/1999 1...	IT_PROG	4200.00
	108	Nancy	Greenberg	NGREENBE	515.124.4569	8/17/1994 ...	FI_MGR	12000.00
	109	Daniel	Faviet	DFAVIET	515.124.4169	8/16/1994 ...	FI_ACCO...	9000.00
	110	John	Chen	JCHEN	515.124.4269	9/28/1997 ...	FI_ACCO...	8200.00



It is important to note that the correct provider for the SQL Server destination is the Microsoft OLE DB provider for SQL Server.

Summary

In this chapter the steps involved in transferring the data in a *Table* in the Oracle 10G XE server to a table in the SQL Server 2005 , were described. Installing and reviewing the objects in the Oracle 10G XE's hr database was also looked at.

14

Web Service Task to Convert Miles to Kilometres

Primarily, a web service is a web application (a software program) that lends itself easily for exchange of data with other web applications. In doing so, it helps businesses to share assets with customers, vendors, etc., and allows them to leverage data in disparate systems.

Some of its features are:

- It adheres to Open Standards like XML, SOAP, TCP/IP, HTTP, etc.
- It is self-contained and modular.
- It uses a standard XML messaging system.
- It is both a programming language and an operating system (Platform) agnostic, as it is based on XML.
- The web service is described by WSDL, and WSDL is discovered by UDDI.

For details about web services, the best source is W3C (<http://www.w3.org/2002/ws/>). The Web Service Task is a new task in SSIS and did not have a counterpart in DTS.

In order to follow the steps as indicated, you will need a web service and the required information to access the service that is usually called the WSDL [Web Services Description Language]. WSDL completely describes the web service that can be accessed over the Internet. In this exercise, the web service will be accessed over the intranet whose URL is <http://localhost>. This URL is that of the default IIS web server, which is assumed to be present. Readers will benefit from reading the web service articles, written from different perspectives, whose links are found in this blog (<http://hodentek.blogspot.com/2007/02/links-to-my-articles-on-web-services.html>).

Hands-On Exercise: Creating and Testing a Package that Uses a Web Service Task

This exercise consists of two parts, the first part describes how the web service is created and how the associated WSDL file that is necessary for the Web Service Task was obtained. In the second part, configuration of the Web Service Task in the SSIS will be described. Both the web service as well as the Business Intelligence Package will be parts of a Visual Studio 2005 solution.

Part One

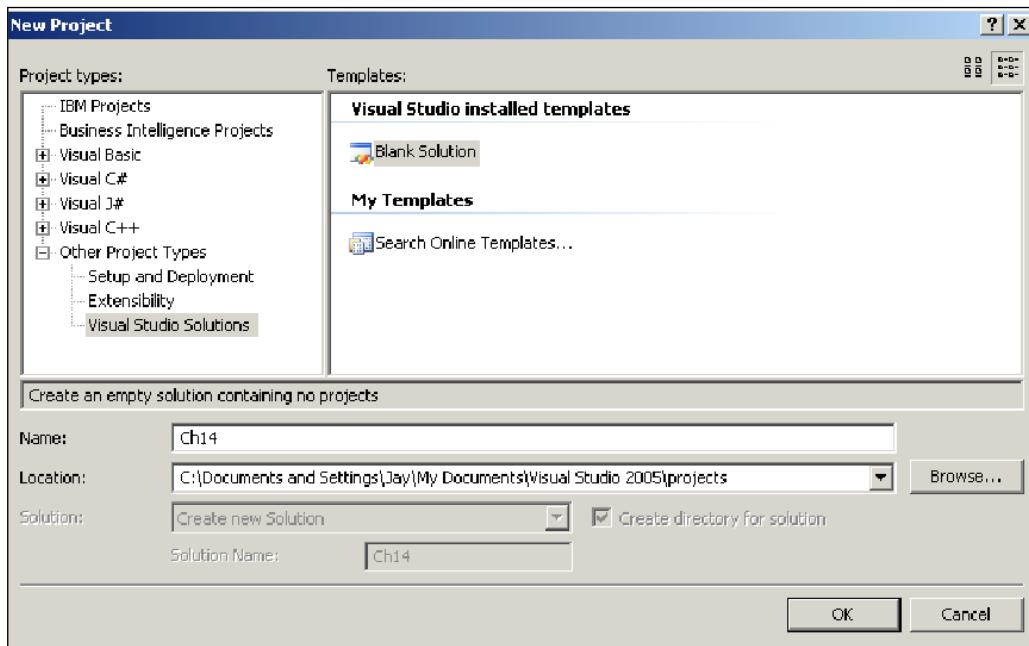
In this part, a web service that converts miles into kilometers will be described. This program, that is going to be hosted on the localhost, can be accessed using the HTTP protocol.

Step 1: Create a Visual Studio 2005 Blank Solution

In this chapter, we will be creating more than one type of project in a VS 2005 solution.

1. Create a solution Ch 14 from **File | New | Project**.

This opens the **New Project** window, as shown in the following screenshot.

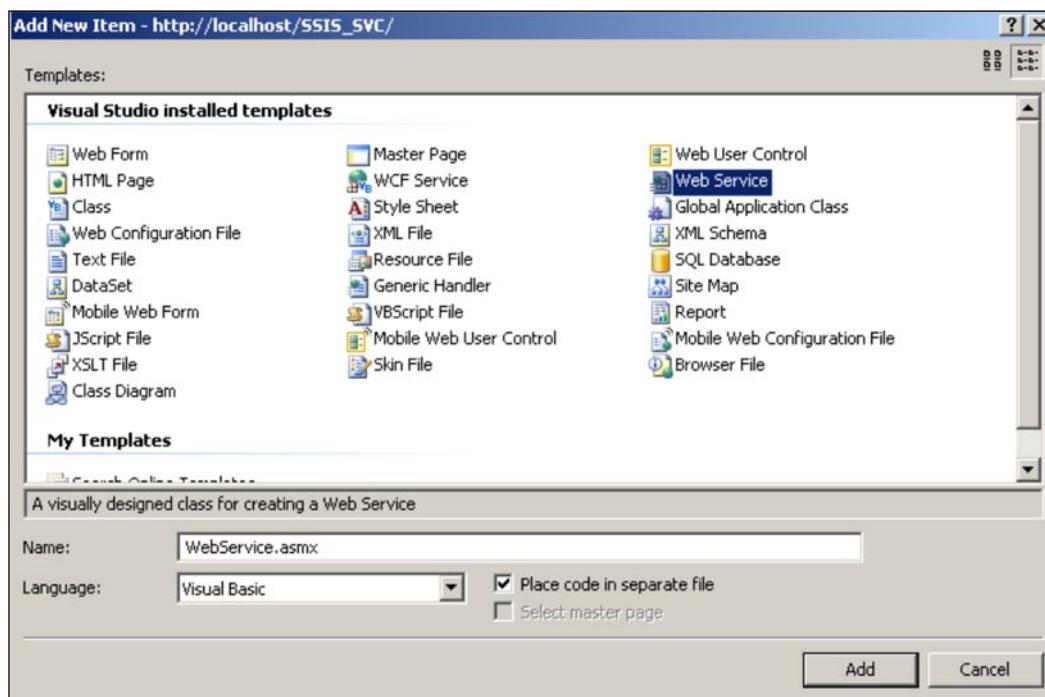


2. Click on the **Other Project Types** and in **Visual Studio installed templates** choose **Blank Solution**.
3. Change the **Name** field to Ch 14.

A Visual Studio blank solution Ch 14 will be created.

Step 2: Create a Web Service

1. From **File | Add | New Web Site...** create a new ASP.NET website **SSIS_SVC** on the localhost.
This adds the site **http://localhost/SSIS_SVC/** to the **Solution**.
2. Right-click after highlighting **http://localhost/SSIS_SVC/** in the **Solution**.
3. In the drop-down, click on **Add New Item...**
This opens the **Add New Item** window, as shown below.

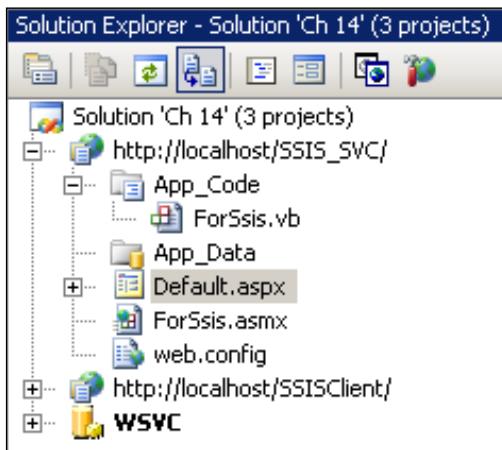


4. Highlight **Web Service** in the Visual Studio installed templates.

This adds a default name **WebService.asmx**.

5. Change the name to `ForSsis.asmx` and click on the **Add** button.

This adds a `ForSsis.vb` file to the `App_code` folder of the `http://localhost/SSIS_SVC` website and a `ForSsis.asmx` file to the website, as shown in the following screenshot.



6. Double-click the `ForSsis.vb` file and substitute the default method shown in the next listing by a new method, so that the complete listing of `ForSsis.vb` is as shown in the following code:

```
<WebMethod()> _
Public Function HelloWorld() As String
    Return "Hello World"
End Function

Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols

<WebService (Namespace:="http://tempuri.org/")> _
<WebServiceBinding (ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class ForSsis
    Inherits System.Web.Services.WebService

    <WebMethod()> _
    Public Function Miles_to_Km(ByVal a As Double) As Double
        Return (a * 1.6093)
    End Function

End Class
```

The file `ForSsis.vb` has the VB code for the web service and this contains the web method. When the client accesses the web service, the function **Miles_To_Km** gets called using the argument (Miles) provided by the web service client, or simply the client. The return value of this function is the value of miles in kilometers.

The web service can be tested by browsing the file `ForSsis.asmx`, which displays the following information. Only a part of the screen is displayed.

The following operations are supported. For a formal definition, please review the [Service Description](#).

- [Miles to Km](#)

This web service is using <http://tempuri.org/> as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. <http://tempuri.org/> is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

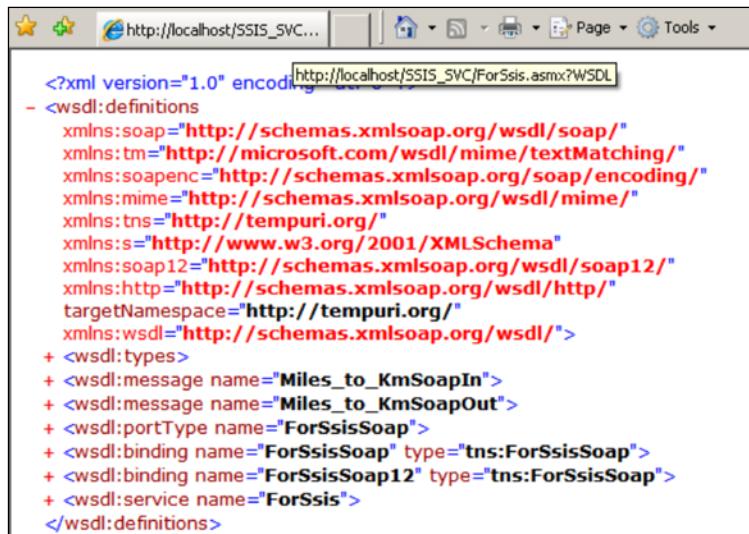
You can invoke the web service by clicking on the **Miles_to_Km** hyperlink and providing an argument to the function in the window that pops-up.

Step 3: Create a WSDL File

The **Service Description** hyperlink points the URL reference (`http://localhost/SSIS_SVC/ForSsis.asmx?WSDL`) to the WSDL by displaying an XML file that completely describes the web service.

1. Click on the **Service Description** hyperlink after browsing to the `ForSsis.asmx` file in the Solution Explorer using Internet Explorer.

The browser display of the file, with several nodes in collapsed state, is shown in the next screenshot.



A screenshot of a Microsoft Internet Explorer browser window. The address bar shows the URL: `http://localhost/SSIS_SVC/ForSsis.asmx?WSDL`. The main content area displays the WSDL (Web Services Description Language) XML code. The XML is well-formatted with color-coded namespaces and collapsible sections indicated by '+' and '-' symbols. Key parts of the XML include the `<wsdl:definitions>` section which defines the service, port type, and various message types like `Miles_to_KmSoapIn` and `Miles_to_KmSoapOut`.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="ForSsis" targetNamespace="http://tempuri.org/ForSsis" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/ForSsis" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <wsdl:message name="Miles_to_KmSoapIn">
      <wsdl:part name="parameters" type="tns:Miles_to_KmSoapInType"/>
    </wsdl:message>
    <wsdl:message name="Miles_to_KmSoapOut">
      <wsdl:part name="parameters" type="tns:Miles_to_KmSoapOutType"/>
    </wsdl:message>
  </wsdl:types>
  <wsdl:portType name="ForSsisSoap">
    <wsdl:operation name="ConvertMilesToKilometres">
      <wsdl:input message="Miles_to_KmSoapIn" />
      <wsdl:output message="Miles_to_KmSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ForSsisSoap" type="tns:ForSsisSoap">
    <wsdl:operation name="ConvertMilesToKilometres">
      <wsdl:input message="Miles_to_KmSoapIn" />
      <wsdl:output message="Miles_to_KmSoapOut" />
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ForSsis">
    <wsdl:port name="ForSsisSoap" binding="ForSsisSoap">
      <wsdl:address>http://localhost/SSIS_SVC/ForSsis.asmx</wsdl:address>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

The web service task editor in the SSIS designer would return an error if one were to use the URL web reference shown in the previous screenshot. The reason for this is that the Web Service Task user interface is expecting a reference to a folder location on the machine. There are a couple of ways to generate this file. The easiest would be to save the source view of the above browser display to a location of your choice with the extension WSDL.

2. In the IE browser click on **View | Source**.

This opens up Notepad displaying an XML file.

3. Save the file (by renaming the source file to `ForSsis.wsdl`) to a location of your choice with the extension WSDL.

In this exercise, it is saved to `C:\Inetpub\wwwroot`.

Part Two

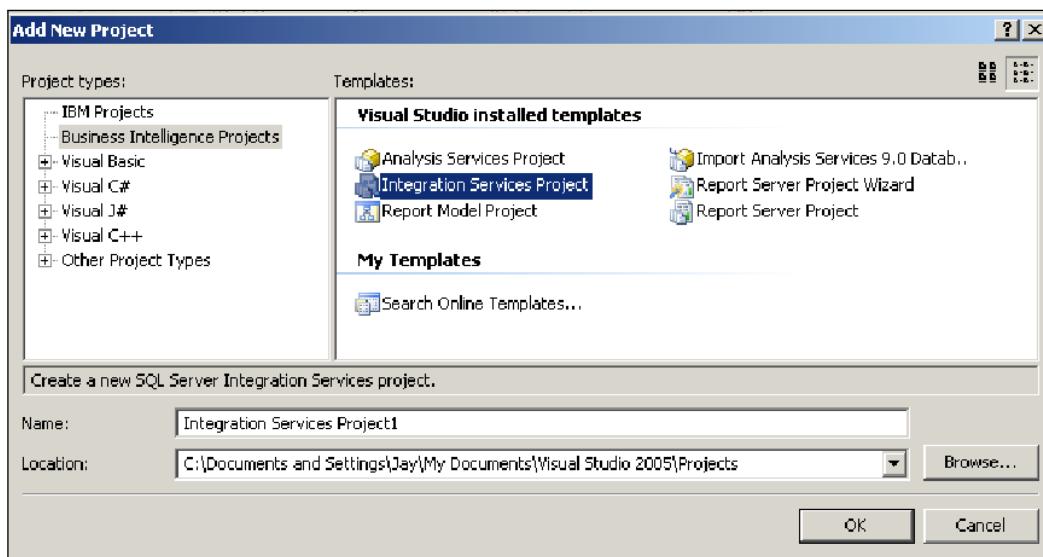
The following steps will be described in setting up a BI project that creates a package containing a Web Service Task. The Web Service Task will call the Web Service created in Part one.

- Creating a BI project and adding a Web Service Task.
- Configuring the Web Service Task.
- Building the BI project and executing the package.

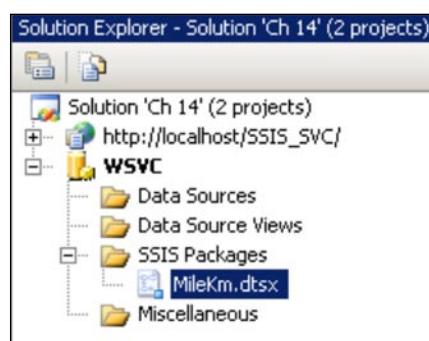
Step 1: Creating a BI project and Adding a Web Service Task

1. Right-click the **Ch 14** solution in the explorer and from the drop-down choose **Add | New Project...**

This opens the **Add New Project** window where you choose to add a SSIS project, as shown in the following screenshot.



2. Rename the project. In this exercise, it was renamed **WSVC** and the default package **.dtsx** was renamed **MileKm.dtsx**, as shown below.



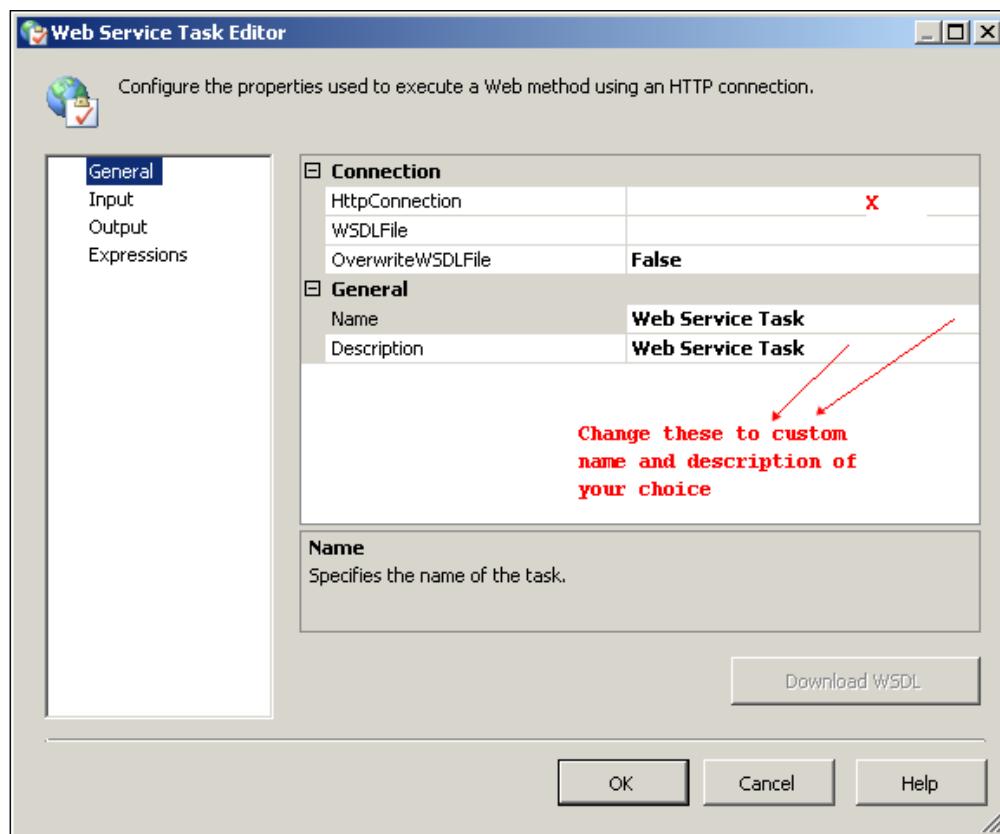
3. Drag and drop a **Web Service Task** from the **Control Flow Items** group in the **Toolbox** onto the **Control Flow** page of the Canvas.

The default name of this Web Service Task will be changed while editing this task in the next step.

Step 2: Configuring the Web Service Task

1. Right-click the **Web Service Task** component and from the drop-down click on **Edit...**

This opens the **Web Service Task Editor**'s window.



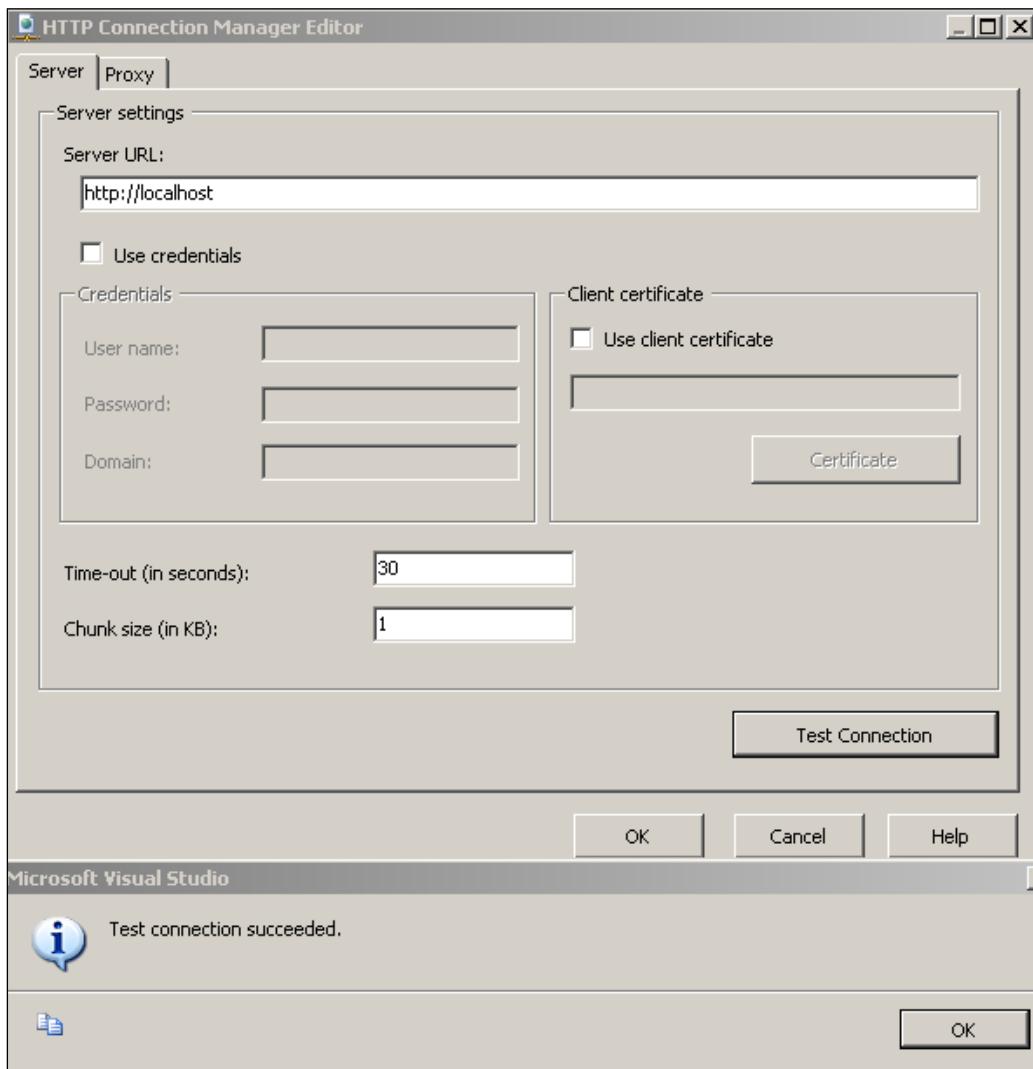
This window has four list items in the left and each of them open their own interactive settings window on the right. For the list item **General**, the following information needs to be given:

HTTP Connection → The protocol used to access the service.

WSDL file → Web Services Description Language, an XML formatted file that describes network services as endpoints for messages. (The process of deriving this file was described earlier).

2. Click on an empty area in the **HTTP line** item at a location **x**, as shown. From the drop-down click on <New connection...>.

This opens the HTTP Connection Manager's window. The HTTP will be requesting directly from the intranet, `http://localhost`.



3. Type in **http://localhost**, as shown. Accept the other defaults and click on **Test Connection** button.

You will get a **Test Connection succeeded** message as the web service is created on the localhost.

The full URL of the web service `http://localhost/SSIS_SVC/ForSsis.asmx` or `http://127.0.0.1` can also be used. For other accessible web services, the complete URL of the web service should be provided. For services on the Internet, you may look up the UDDI registry. This is outside the scope of this exercise.

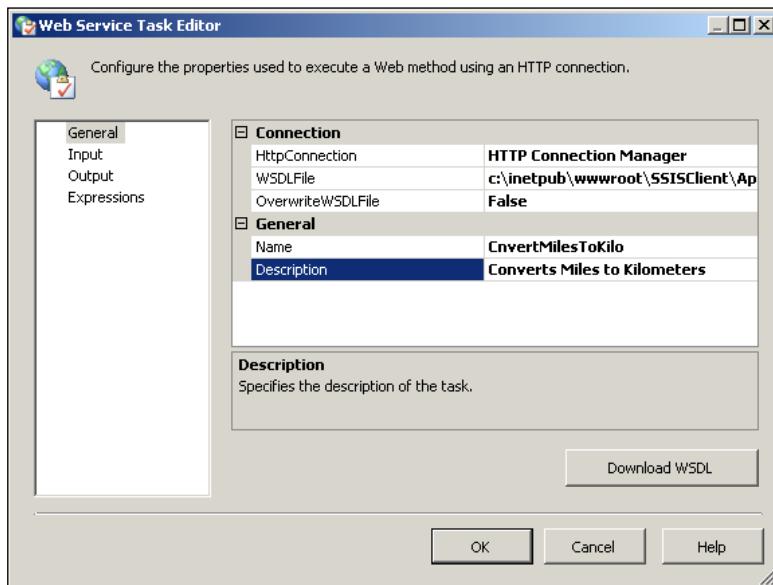
4. Click on the **OK** button on the **Microsoft Visual Studio**'s message window as well as the **HTTP Connection Manager**'s window.

A **HTTP Connection Manager**'s component will be added to the **Connection Managers**' window in the Canvas.

5. Fill in the details (as shown) for the rest of the line items so that the resulting window display is as shown in the following screenshot. For the WSDL file, use the location that was used for saving the source view of the browser display.

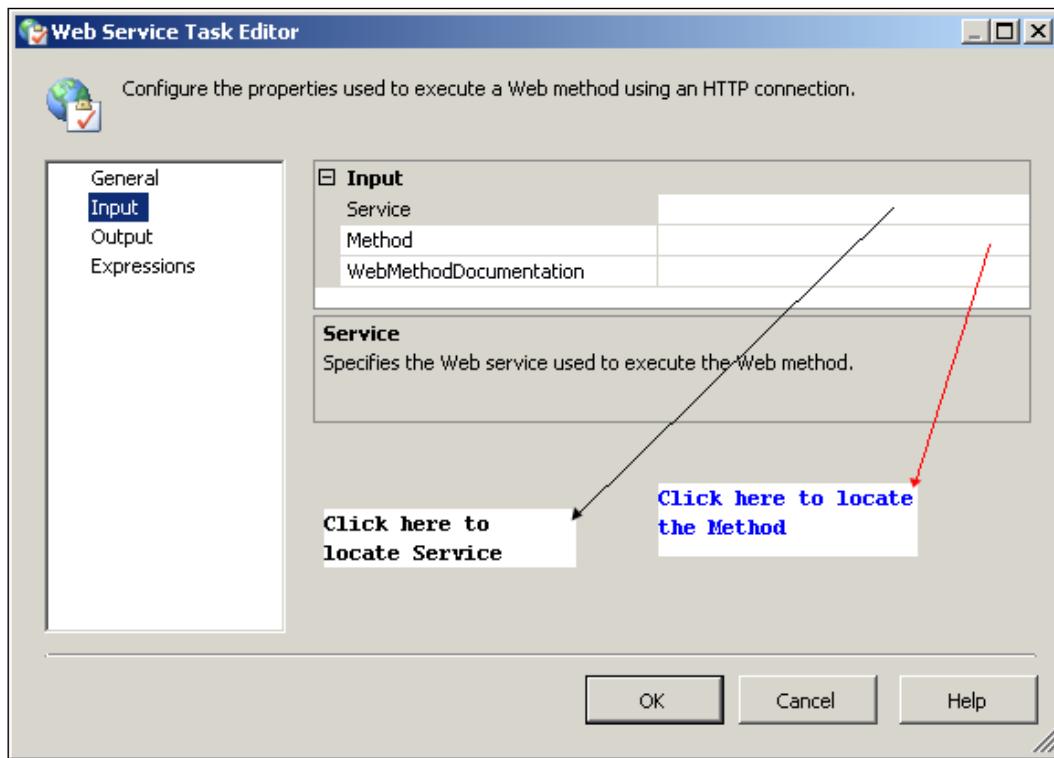
This path is: `c:\inetpub\wwwroot\ForSsis.wsdl`, since the file was saved to this location, as mentioned earlier.

Note that although there is a **Download WSDL** button, it does not get enabled unless you enter the full location above. It is supposed to download the file when the HTTP source is specified. However, it does not work as intended.



6. Click on the list item **Input** in the **Web Service Task Editor**.

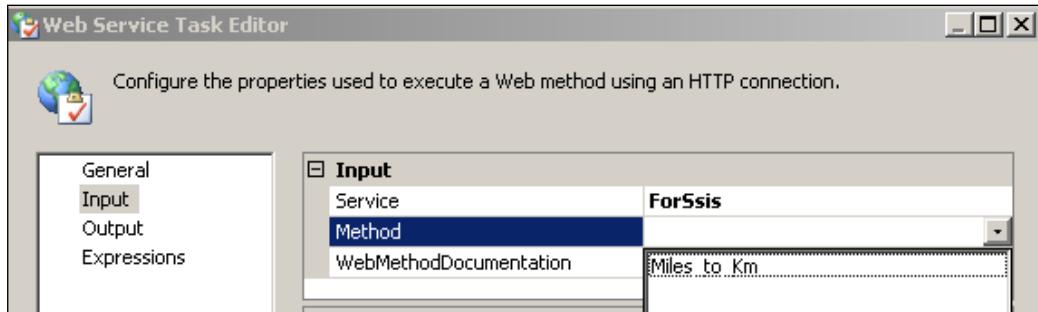
In this window, you need to provide the name of the web service and the method call that you want to make. If the previous window is configured with the correct HTTP connection and a correct WSDL file, then when you click on an empty area along this line item, **ForSsis** should show up. When you click on an empty area for method, the **Miles_To_Km** method should get displayed.



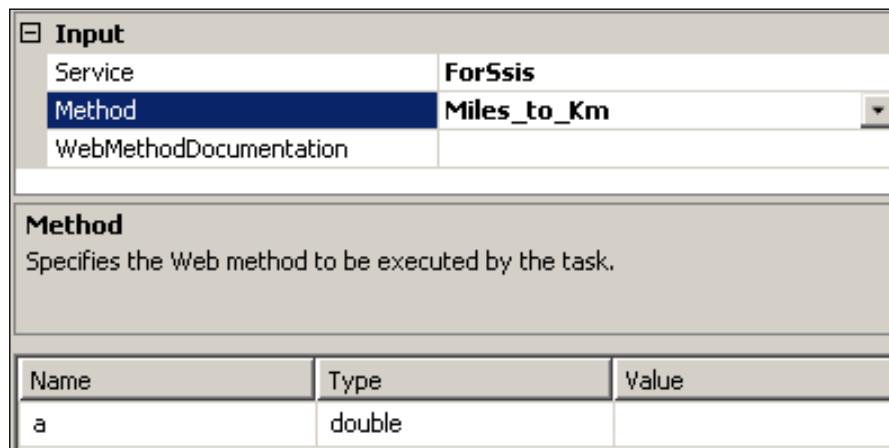
Web Service Task to Convert Miles to Kilometres

7. Click on an empty area to locate the **Service** and click on an empty area to locate the **Method**, as shown in the following screenshot.

These two actions will locate the web service and the method call that were referenced in the WSDL file. In the **WebMethodDocumentation** line item, you provide a verbose description of what the method would do when called in the package.

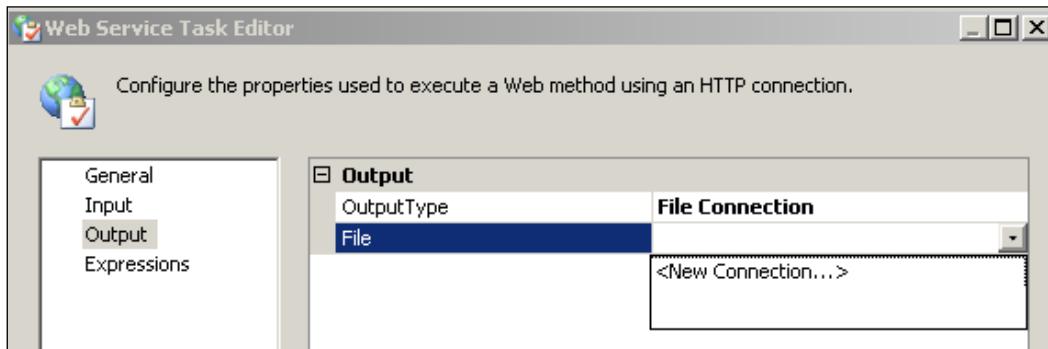


8. When you choose the **Miles_To_Km** web method, the method's input argument window opens. Just type in a value (say 20) for the argument **a** in the box under the column heading **Value**.



9. Click on the list item **Output** on the left.

This brings up the window shown below.



This window will open a **File System Task Editor** when you click on the line item **File** (on an empty area), which we have seen earlier many times. In the **File System Task Editor** choose to create a new file *WebServiceTaskOutput* or any file name of your choice.

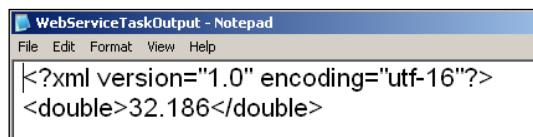
10. Click on the button **OK** on the Web Service Task Editor.

The Web Service Task is now completely configured.

Step 3: Building the BI project and Executing the Package

1. Right-click the **WSVC** project in the **Solution Explorer** and click on **Build** in the drop-down menu. After the Build succeeds, click on the *MileKm.dtsx* and execute the package.

After the package has run the **CnrvrtMilesToKilo** web service, the task turns green and will have created the file *WebServiceTaskOutput*. The next screenshot shows a display of the file when opened with Notepad. The file was not given any extension, but it happens to be an XML file. The result of converting 20 miles yields 32.186 kilometers.



Summary

This chapter described the usage of the Web Service Task in the SSIS Editor. A simple web service was described. Compared to other chapters, the Visual Studio 2005 Solution consists of two parts, both of which were shown in detail. It is recommended that you review some of the articles on web services, whose links have been provided, as well as get a working knowledge of the scope and extent of web services from the W3C site. It would have been a lot easier (user friendly) if the tool would take a URL reference rather than a file reference.

15

Package that Transfers a Database from One SQL Server to Another

Database migration from one version of SQL Server to another version, database consolidation, and addressing availability issues by duplication to another remote standby location are some of the scenarios where you need to transfer or copy a database.

When you need to transfer (move) databases between two SQL Servers or make a copy of a database on the same SQL Server, you would be using the Transfer Database Task. For migrating databases from an earlier version (SQL Server 2000) to SQL version 2005 as well, you would need this task, though there are other methods available for the transfer.

There are two ways in which databases can be transferred using the Transfer Database Task: the online and offline modes.

In the online mode, the transfer takes place through the mediation of the SQL Management Object (SMO) connections.

In the offline mode, the transfer takes place by detaching and attaching the database files, which render the database unusable during the transfer, and also requiring SMO connections. Also, in order to successfully transfer a database from one server to another, you must have proper permissions for the databases, as well as files and folders on the two computers.

The Transfer Database Task also permits overwriting a destination database with the same name replacing the contents. There are some restrictions on the size of the database i.e. to be transferred relative to the size of the *model* database, as discussed in this online informative article, <http://msdn2.microsoft.com/en-us/library/ms141204.aspx>, which describes this task in greater detail.

Further, the Transfer Database Task component in SSIS is error prone as it assumes only one installed SQL Server, which may not be the case in general. In this case, one has to be careful about which of the servers the Task Editor brings up automatically during configuration.

Hands-On Exercise: Creating and Testing a Package that Uses a Transfer Database Task

In order to follow the steps as indicated, you will need a Transfer Database Task, the ability to make connection to the two servers, and the required permissions. In this exercise, the **pubs** database on the SQL Server 2000 on a networked computer, *Computer 1 (XPHTEK in this example)*, will be copied over to the SQL Server 2005 Express on the *Computer 2 (Hodentek in this example)*, on which the Visual Studio 2005 is also installed. If **pubs** are not available for some reason, any other database may be used. In addition to the SQL Server 2005 Express, Computer 2 also has an SQL Server 2005 Standard Edition installed.

Step 1: Creating a Network Share on Computer 1

As a preparation to the Transfer Database Task, a network share on Computer 1 will be created, because this share will be needed during the transfer.

1. Create a network share (called **PubsData**) on Computer 1, so that the Data folder of SQL Server 2000 is accessible from Computer 2.

This folder will be normally found at the location `C:\Program Files\Microsoft SQL Server\MSSQL` for a default installation.

The following steps will be described in setting up a BI project that creates a package containing a Transfer Database Task.

- Creating a BI project and adding a Transfer Database Task.
- Configuring the Transfer Database Task.
- Building the BI project and executing the package.

Step 2: Creating a BI Project and Adding a Transfer Database Task

1. Create a Business Intelligence Project Ch 15 and rename the default name of the package.

The default name was changed to TransferDb.dtsx in this exercise.

2. Drag and drop a **Transfer Database Task** from the **Control Flow Items** group in the **Toolbox** to the **Control Flow Page** on the **Canvas**.

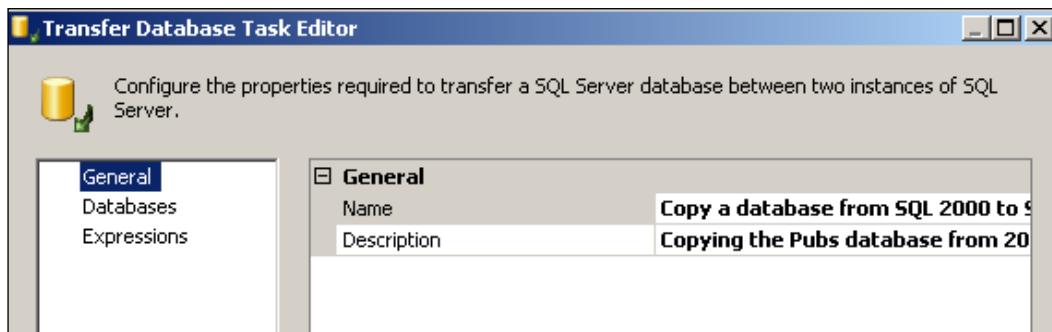
Step 3: Configuring the Transfer Database Task

1. Right-click the **Transfer Database** component in the **Control Flow Page** on the **Canvas** and from the drop-down choose **Edit...**

The **Transfer Database Task Editor** opens displaying the **Name** and **Description** properties of the **General** tab.

2. Change the **Name** and **Description** properties to suitable ones for this exercise.

The name was changed from **Transfer Database Task** to **Copy a database from SQL 2000 to SQL 2005** and the description from **Transfer Database Task** to **Copying the Pubs database from 2000 to 2005 Server** as shown in the following screenshot (partial view shown).



3. Click on the **Databases** list item on the left.

This opens the window where information about the two servers involved in the transfer must be provided. The main details to be furnished are shown in the partial view, as shown in the next screenshot.

Connections	
SourceConnection	1
DestinationConnection	2
Destination Database	
DestinationDatabaseName	
DestinationDatabaseFiles	
DestinationOverwrite	False
Source Database	
Action	Copy
Method	DatabaseOffline
SourceDatabaseName	
SourceDatabaseFiles	
ReattachSourceDatabase	False

There are three main areas that need to be considered.

In the **Connections** node, the connection information necessary to connect to the **Source** and the **Destination** needs to be furnished.

In configuring the **Source Database** node, several possibilities exist as the **Action** item can be *Copy* or *Move*, and the **Method** item can be *DatabaseOffline* or *DatabaseOnLine*. Also, if the method chosen is *DatabaseOffline*, the **SourceDatabaseFiles** item describing the name and location of database files needs to be furnished along with whether or not the **SourceDatabaseFile** needs to be reattached.

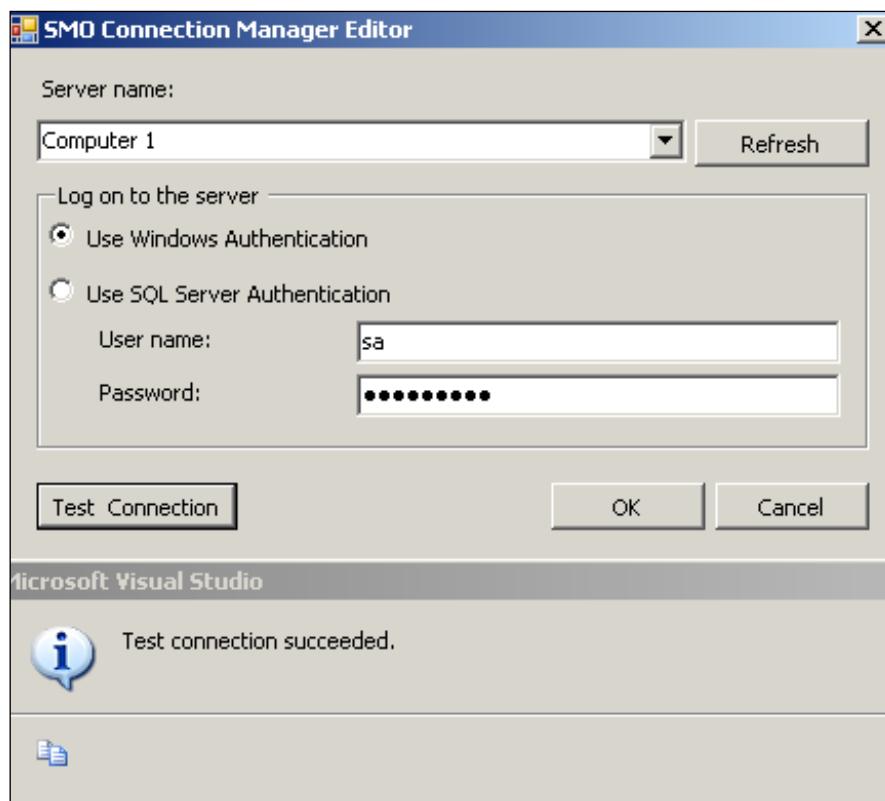
In configuring the **Destination Database**, information similar to configuring the Source Database should be provided, in addition to providing whether or not the destination database is to be overwritten.

In this exercise, we will be considering the **DatabaseOnline** method and the action to **copy** the database.

Establishing Connections

1. Click on the point marked **1** along the **SourceConnection** item, shown in the previous screenshot.
2. From the drop-down menu, click on <New connection...>.

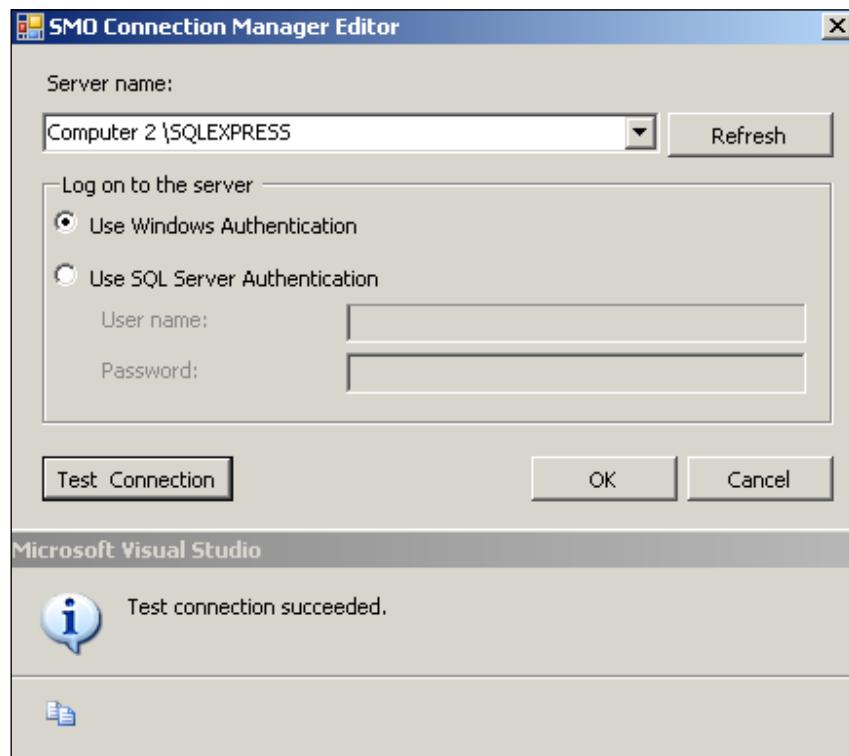
This brings up the **SQL Management Object (SMO) Connection Manager Editor** window as shown in the next screenshot. When this window shows up, all the controls are empty. The database pubs is on the SQL Server 2000 on the machine Computer 1. For the server name field, type-in **Computer 1 (XPHTEK)**, as shown in the next screenshot (you may also pick it from the drop-down menu when you click on the drop-down arrow head). This server uses the SQL Server authentication; type-in the **User name** and **Password**, as shown (you need to use those appropriate for this networked server). The connection can be tested as well.



3. After the Test succeeds, click the **OK** button on the **Microsoft Visual Studio 2005** window and the **SMO Connection Manager Editor** window.

This completes the connection to the source and a **SMO Connection Manager** component is added to the **Connection Managers**' page of the Canvas.

4. Click on location 2, along the destination connection, and as done in the previous case, enter server name and authentication information (Windows authentication this time) as shown in the next screenshot. Note that the fully qualified name of this server is *Computer 2 \ SQLExpress (Hodentek/SQL Express)*.



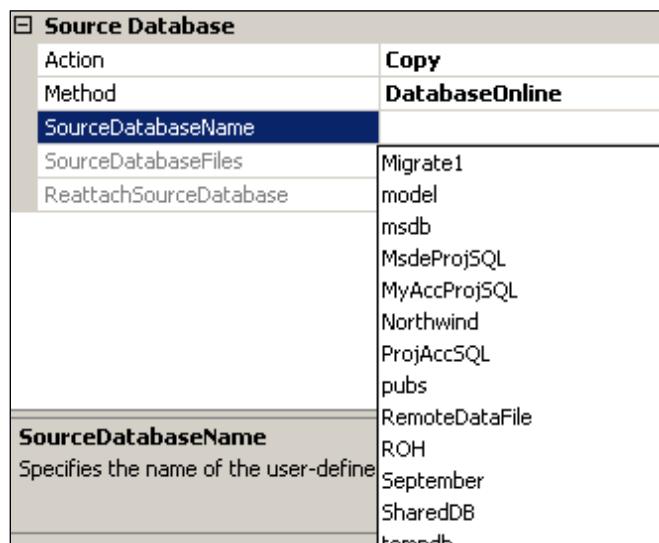
5. Click on the OK button on the **Microsoft Visual Studio** window, as well as the **SMO Connection Manager Editor** window.

This adds another **SMO Connection Manager** control to the Connection Managers' page on the Canvas. The **Connections** window of the **Databases** list item of the **Transfer Database Task Editor** now gets changed, as shown below.



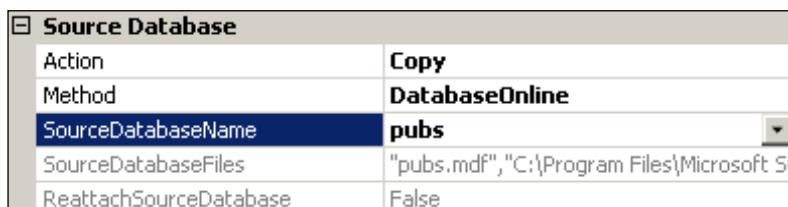
Configuring the Source Database Node

1. In configuring the **Source Database**, choose **Copy** for the **Action** item from the drop-down and choose **DatabaseOnline** from the **Method** item's drop-down. This grays out the **SourceDatabaseFiles** and the **ReattachSourceDatabase** items.
2. Click on the **SourceDatabaseName** drop-down handle to bring up the list of databases on the SQL 2000 Server, as shown in the following screenshot. You may verify if a database **pubs** does exist on this server. Although **pubs** is chosen from this list, you can use any other database that exists on your server.



3. Click on **pubs** in the drop-down list.

This updates the information in the **Transfer Database Task Editor** window. Although the **SourceDatabaseFiles** was grayed out, the file name and location name are still retrieved. At present, the information is read-only. The grayed out information is incomplete as it was mentioned that this task is error prone (perhaps for this version). Follow the steps to retrieve the complete information.



4. Change the method to **DatabaseOffline** by clicking along the line item, and choosing it from the drop-down menu.

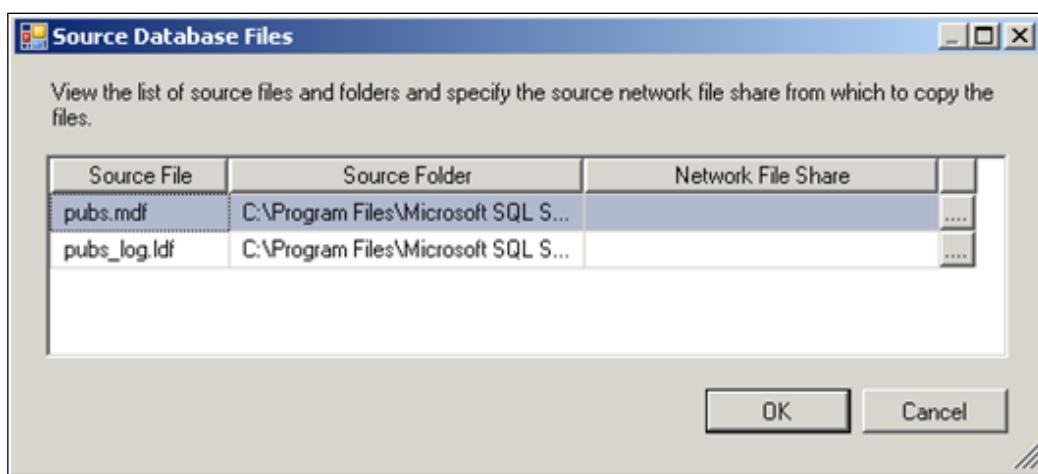
This enables the **SourceDatabaseFiles** to be modified.

5. Click on an empty area along the **SourceDatabaseFiles**.

This will display an ellipsis button.

6. Click on the ellipsis button

This brings up the **Source Database Files** window.



7. Click on the ellipsis button to the right of the **Network File Share** list heading.

This opens the **Browse for folder** window where you need to locate the shared data folder **PubsData** on Computer 1 you created earlier.

8. Click on **PubsData** and click on the **OK** button on the **Browse the Folder** window.

The UNC name of the share (`\Computer 1\PubsData`) will appear in the **Network File Share** boxes.

9. Click on the **OK** button on the **Source Database Files** window.

10. Now change the **Method** to **DatabaseOnline**.

This inserts the database name for the destination as shown in the next screenshot as well as the DestinationDatabaseFiles related information.

SourceConnection	Computer1.sa
DestinationConnection	Computer 2\SQLEXPRESS
Destination Database	
DestinationDatabaseName	pubs
DestinationDatabaseFiles	"pubs.mdf","C:\Program Files\Microsoft SQL Server\M..."
DestinationOverwrite	True
Source Database	
Action	Copy
Method	DatabaseOnline
SourceDatabaseName	pubs
SourceDatabaseFiles	"pubs.mdf","C:\Program Files\Microsoft SQL Server\M..."
ReattachSourceDatabase	True

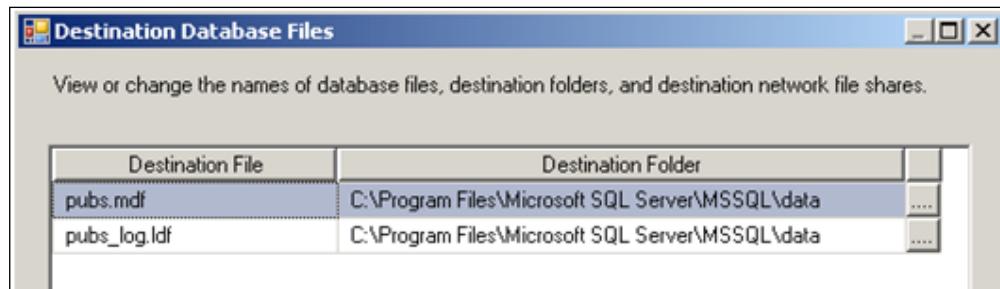
Configuring the Destination Database

The destination database is on the same machine as the Visual Studio on which the package will execute.

Again, the **DestinationDatabaseFiles** item value generated by this task can be error prone and needs to be rectified. The reason is that the **Transfer Database Task** does not correctly identify the SQL Server to which the transfer is to be made when a plurality of SQL Servers are installed on the machine. The reader may refer to the following article (<http://www.aspfree.com/c/a/MS-SQL-Server/Transferring-a-Database-Using-the-SSIS-Designer/>) by the author for more detailed explanation.

1. Click on an open area along the line item **DestinationDatabaseFiles**.

This opens the **Destination Database Files** window, as shown in the following screenshot. You can see that this is not the SQL Server chosen for data transfer.



The correct destination files reference, for data going into the *SQLExpress* server on this machine should be the following (note the highlighted database server name):

```
"pubs.mdf", "C:\Program Files\Microsoft SQL Server\MSSQL.4\MSSQL\data", "", "pubs_log.ldf", "C:\Program Files\Microsoft SQL Server\MSSQL.4\MSSQL\data", ""
```

2. After rectifying the **Destination Folder** information, click on the **OK** button on the **Destination Database Files** window.

This updates the **Destination Database Files** information on the **Transfer Database Editor**.

3. Click on the button **OK** on the **Transfer Database Task Editor**.

This completes the configuration of the **Transfer Database Task**.

Step 4: Building the BI Project and Executing the Package

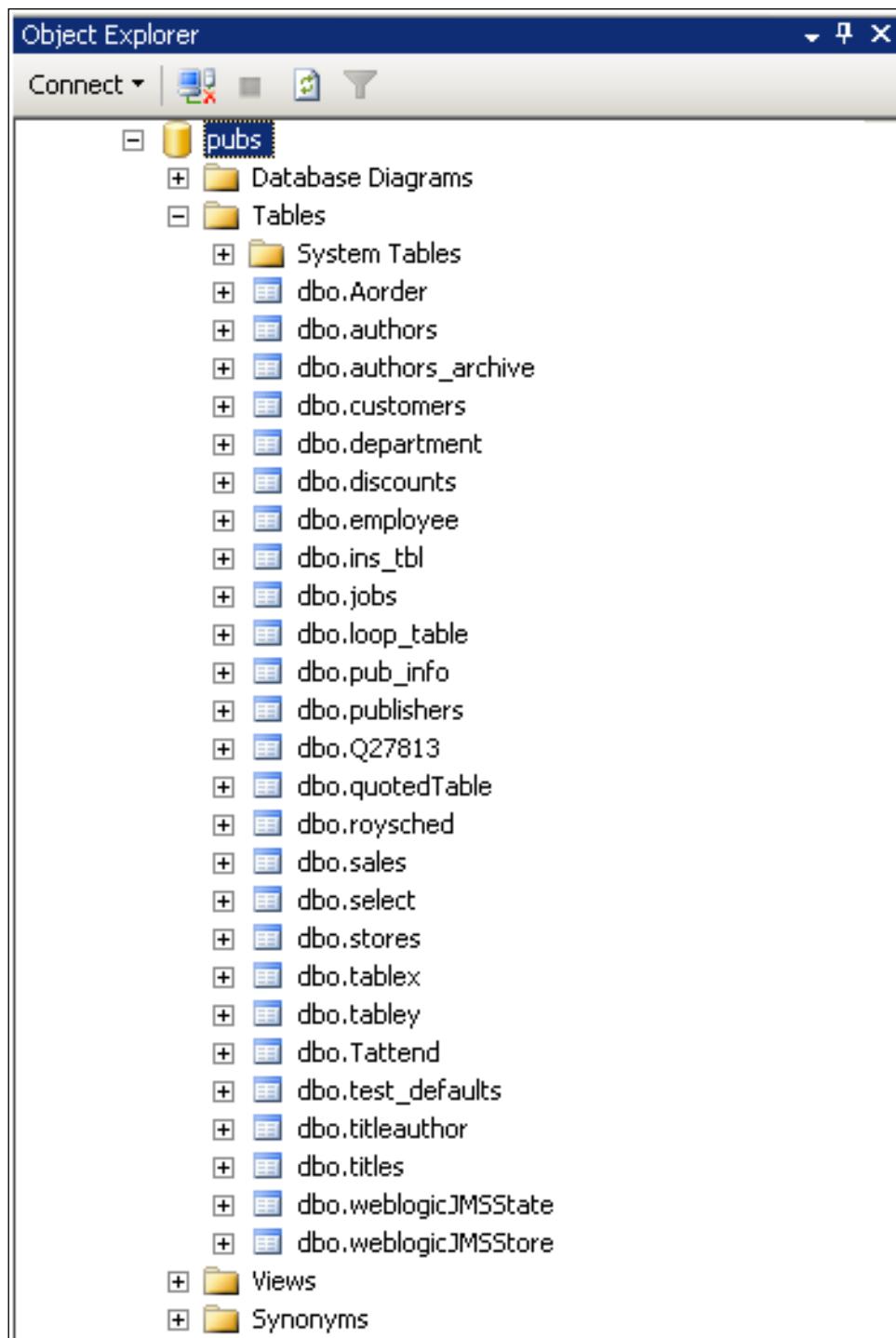
1. Right-click the **Ch 15** project in the **Solution Explorer**, and click on **Build** in the drop-down menu. After the Build succeeds, click on the **TransferDb.dtsx** and execute the package.

After the package is run, the **Create a database from SQL 2000 to SQL 2005** task turns yellow then green, and will have copied the **pubs** database to the SQL Server 2005. This may take a little longer time than in the previous chapter runs.

Now, it is time to verify the success of this task.

2. Open the **SQL Server 2005 Management Studio** and the **SQL Server 2005 Express**, and then expand the **Database** node as shown.
3. Review the database objects by expanding each node where necessary.

You may have to carry out a server refresh to see the **pubs** database. Only the **Table** and **Stored** procedure are shown expanded in the screenshot.



Summary

This chapter described the usage of the Transfer Database Task wherein the pubs database on a networked SQL Server 2000 was copied over to the SQL Server 2005.

Some work arounds were shown for configuring the Source and Destination database items in the editor, as some of the default actions of this editor are error prone. The article did not discuss how well the database was transferred, though it was shown that the entire user created Tables, Stored Procedures, and Views were copied.

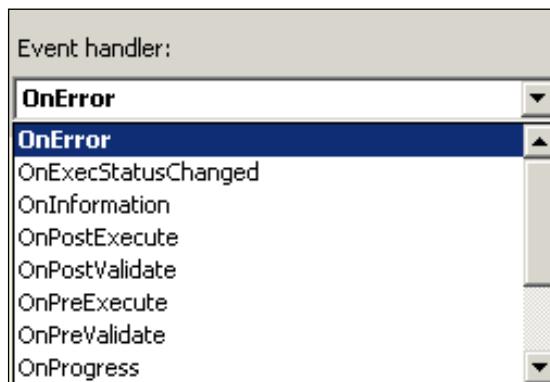
16

On Using Event Handlers in SSIS

Executables such as packages, for loop, task host containers, etc., raise events at run time. The most common event is a task that may fail. This raises an *OnError* event, the default event, when the task fails. These events can be leveraged by handling the event in such a way that some other task can be accomplished or information about the failure gleaned. Thus, by using event handlers the package's functionality is extended.

In this exercise, you will create a package that is sure to fail! When this happens you notify by email that the task did fail using an event handler for the *OnError* event. You will also learn about the *OnPostExecute* event to ping your default gateway to verify that the mail can go out of your machine. Albeit a contrived example, you will also learn about the Execute Package Task and the Execute Process Task.

The SSIS designer comes with a number of standard event handlers, as shown in the following screenshot.



Hands-On Exercise: Creating a Project with Two Packages

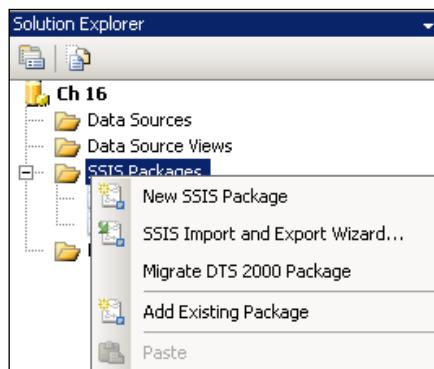
In order to follow the steps as indicated, you will need to create a Business Intelligence project that has two packages. When the first package fails, you will use the OnError event to execute the other package. Simultaneously, when the first package finishes executing you verify that you can successfully ping your default gateway.

The following steps will be described in setting up a BI project with two packages:

- Create a BI project and rename the default package.
- Add and configure the package that has a Send Mail task.
- Add and configure an Execute SQL Task to the renamed default package.
- Add an Execute Package Task to the OnError event of the renamed default package.
- Add an Execute Process Task to the OnPostExecute event of the renamed default package.
- Build and execute the package with event handlers and verify.

Step 1: Create a BI Project and Rename the Default Package

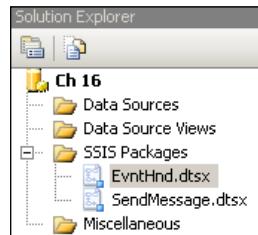
1. Create a Business Intelligence Project **Ch 16** and rename the default file `Package1.dtsx` to `EventHnd.dtsx`.
2. Right-click the **SSIS Packages** folder in the **Solution Explorer** and from the drop-down click on **New SSIS Package**, as shown in the following screenshot.



This adds a `Package1.dtsx`.

3. Rename the file name to `SendMessage.dtsx`.

The Solution Explorer will display two packages, as shown below.



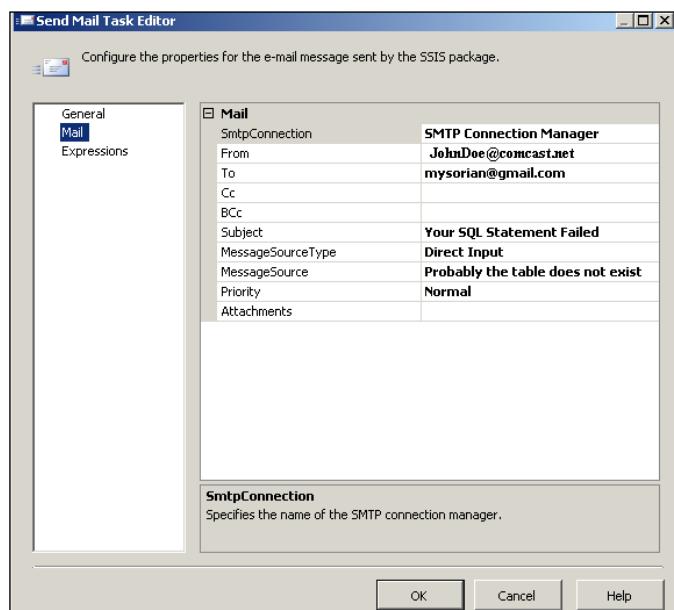
Step 2: Add and Configure the Package that has a Send Mail Task

1. Double-click the `SendMessage.dtsx` in the **Solution Explorer**.

This brings up the Canvas for the `SendMessage.dtsx` package.

2. Add and configure a *Send Mail Task*.

Chapter 3 describes the *Send Mail Task* in great detail. Review Chapter 3 and arrive at the fully configured package as shown in the next screenshot. The **Name** of the task is "**Failure Message**", and the **Description** of the task can be anything meaningful.

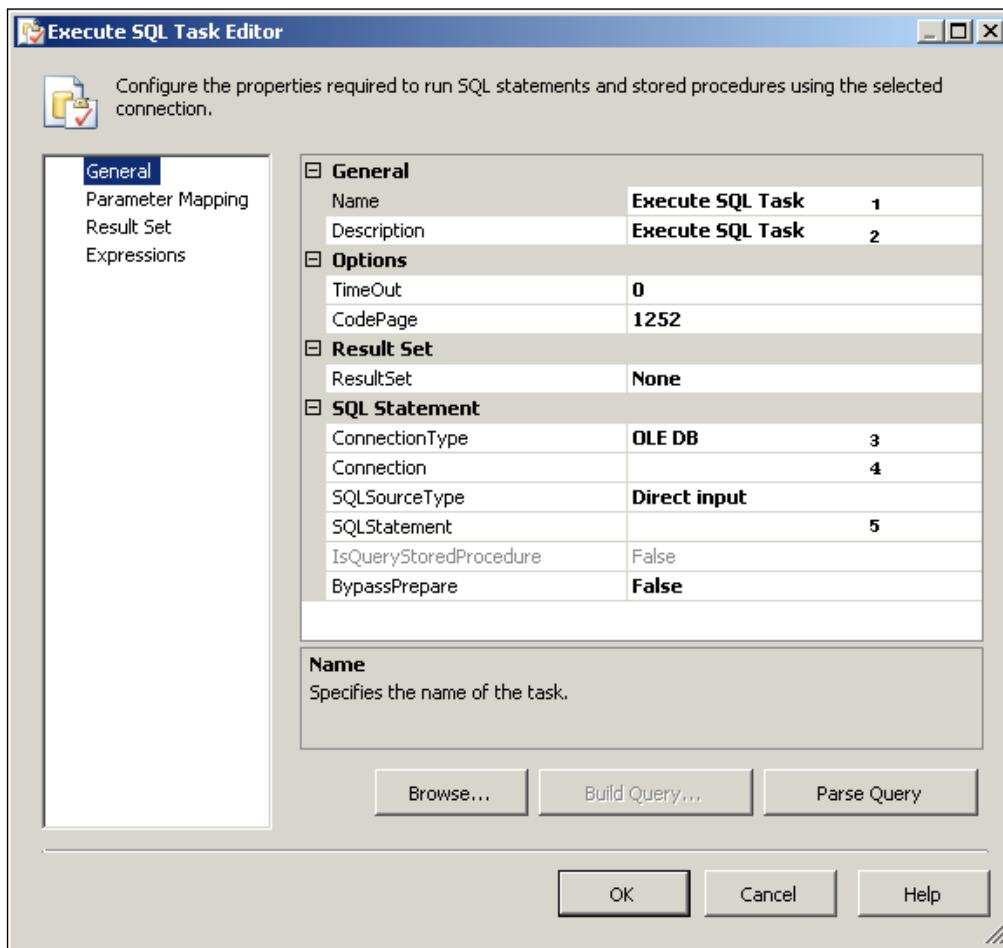


This package may be independently tested and verified.

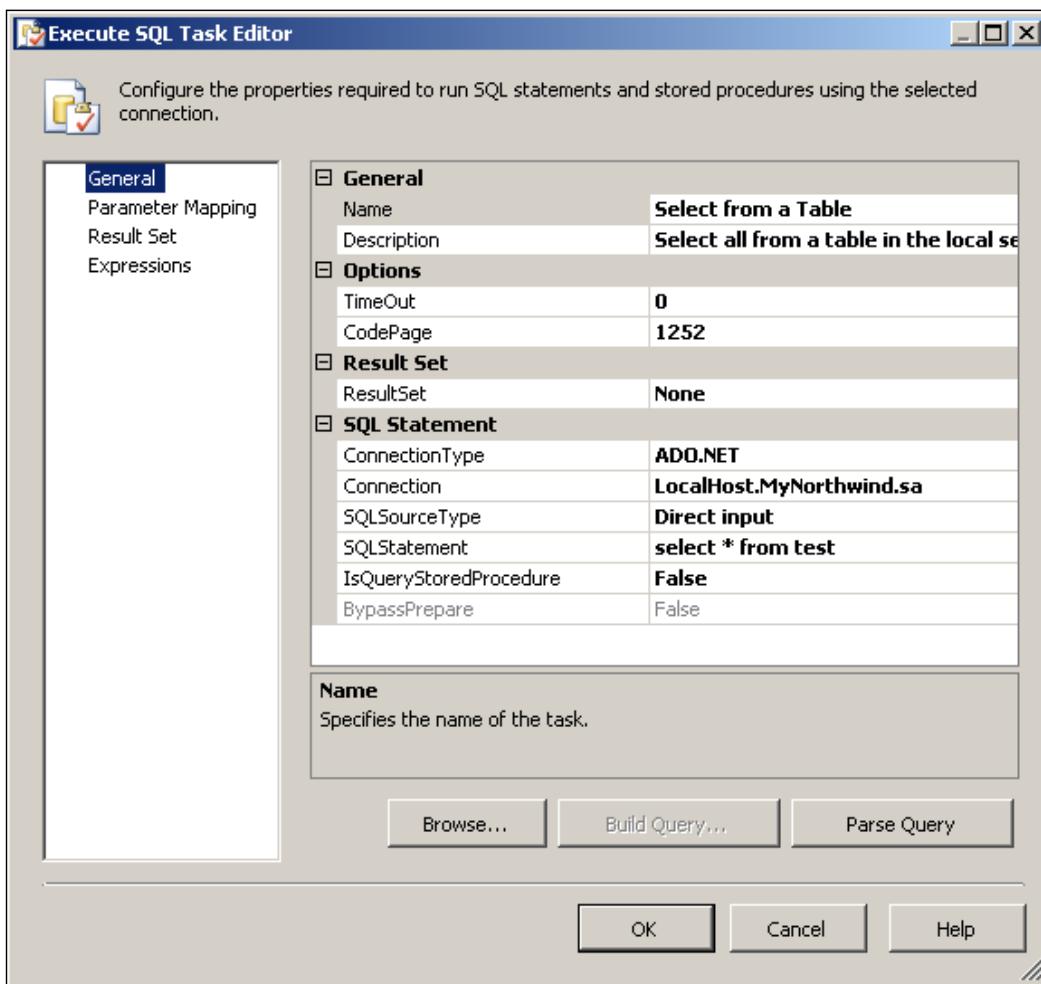
Step 3: Add and Configure an Execute SQL Task to the Renamed Default Package

1. Double-click the **EvntHnd.dtsx** in the Solution Explorer to bring its Canvas into focus.
2. Drag and drop an **Execute SQL Task** from the **Control Flow Items** group in the **Toolbox** onto the **Control Flow** page of the Canvas.
3. Right-click the **Execute SQL Task** component on the Canvas and from the drop-down choose **Edit** (or just double-click the component).

This opens up the **Execute SQL Task Editor**, as shown in the following screenshot (the numbers 1 to 5 are inserted by the author).



4. At points indicated **1** and **2** on the screenshot, change the name and description, as shown below.
5. Click on the point indicated at **3**, and from the drop-down choose **ADO.NET** as the connection type.
6. Click on the point indicated **4**, which will bring up the <New connection...> drop-down menu item, which in turn will bring up the "**Configure ADO.NET Connection Manager Editor**" window.
7. Configure the Connection to connect to your SQL server, like you did in earlier chapters.
8. At point **5**, insert an invalid SQL statement, as shown below.



You may also use the **Browse...** button to look for a stored query on your hard drive, and you could parse them using the **ParseQuery** button. When this component executes, it will go and look for a table called *Test* on the Local SQL Server. As there is no table by this name, the component would fail and generate an error. You may execute and verify that it fails.

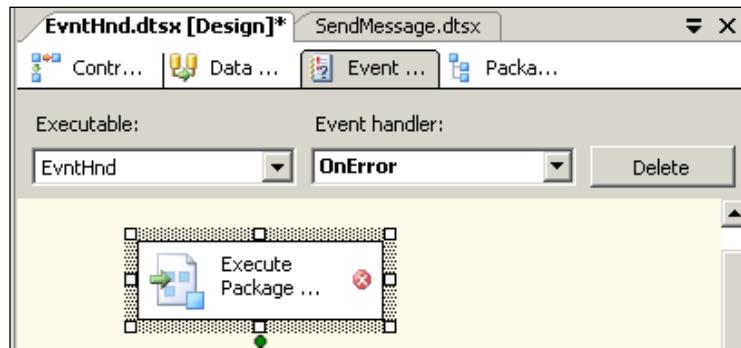
Step 4: Add an Execute Package Task to the OnError Event of the Renamed Default Package

1. Click on the **Event Handler** tab of the *EvntHnd.dtsx* package.

The page opens with the **OnError** Event Handler, as shown in the following screenshot, but with a gray background. Click on the link, **Click here to create a 'OnError' error-handler for package 'EvntHnd'**.

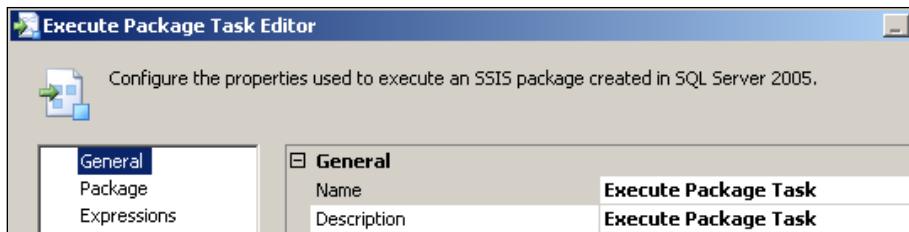
2. Drag and drop an **Execute Package Task**, as shown in the following screenshot.

The Execute Package Task gets executed when the error is raised due to the failed package *EvntHnd.dtsx*.



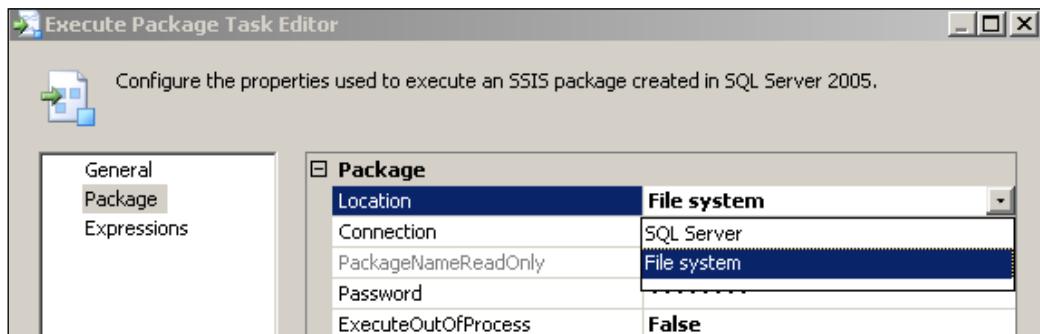
3. Double-click the **Execute Package Task** component.

This opens up the Execute Package Task Editor with display showing the contents of the General list item. You may change them suitably to reflect the scenarios you are planning. Here it was left as is.



4. Click on the **Package** list item on the left.

This will open the details of the package that needs to be executed such as its location (as shown in the following screenshot). This location can be on the SQL Server or the local file system, as shown below.



5. Choose **File system** and on the next line you need to point to a **Connection**. Click on an empty area here and click on the drop-down menu <New connection...>.

In the following **File System Connection Manager Editor** window:

6. Choose the option, an **Existing file** and follow though the folders till you find **Ch 16**, and there in, choose to add the **SendMessage.dtsx** that you created earlier.
7. Leave the other defaults as they are and click on the **OK** button on the **Execute Package Task Editor**.

This completes configuring the **OnError** event handler.

You may now test and verify that when the **EvntHnd.dtsx** is executed, a message will be sent to your email address as this package will generate an error (non-existent table). Here, everything is hard coded.

Step 5: Add an Execute Process Task to the OnPostExecute Event of the Renamed Default Package

Let us take a look at adding an event handler to the **OnPostExecute** event. This event is raised as soon as `EvntHnd.dtsx` finishes executing.

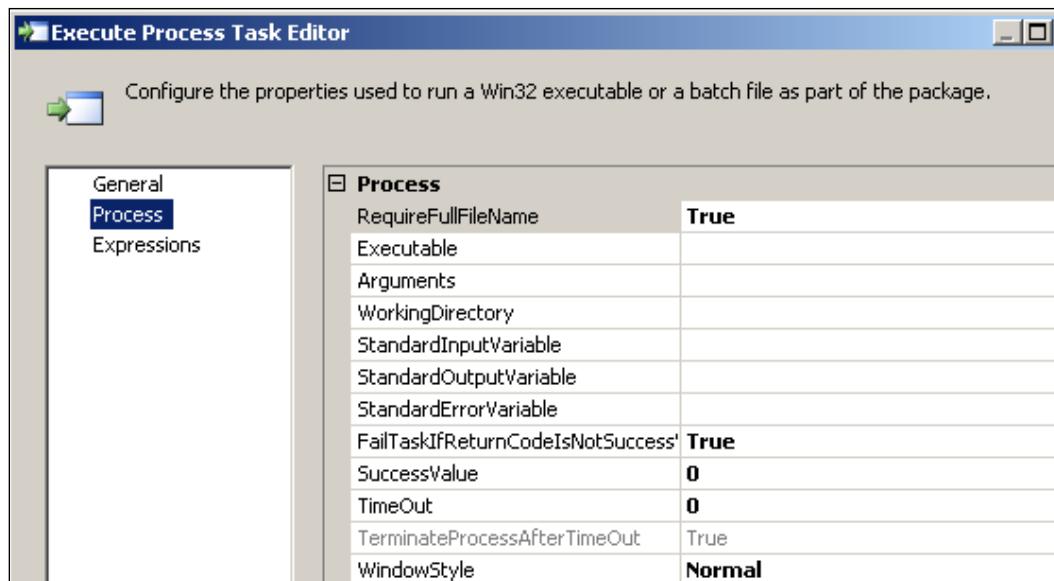
1. In the canvas of `EvntHnd.dtsx`, click on the **Event Handler** tab and choose from the drop-down the event **OnPostExecute**.

Choose to add an event handler by clicking on the hyperlink on the gray background (just like you did earlier for **OnError** event).

2. Drag and drop an **Execute Process Task**, a task when executed will bring up a process (an executable such as `Notepad.exe`, `Calc.exe`, etc.) and then double-click this component after placing it on the Event handler page of the Canvas.

This brings up the **Execute Process Task Editor** page with the default list item **General** displaying the Name and Description of the task. You can rename them. Here the **Ping the Default Gateway** is used for the **Name**. For the **Description** **Check if default gateway is accessible** was used.

3. Click on the **Process** list item, which opens up the window where in you need to supply information as to what process you want it to run, as shown in the following screenshot.



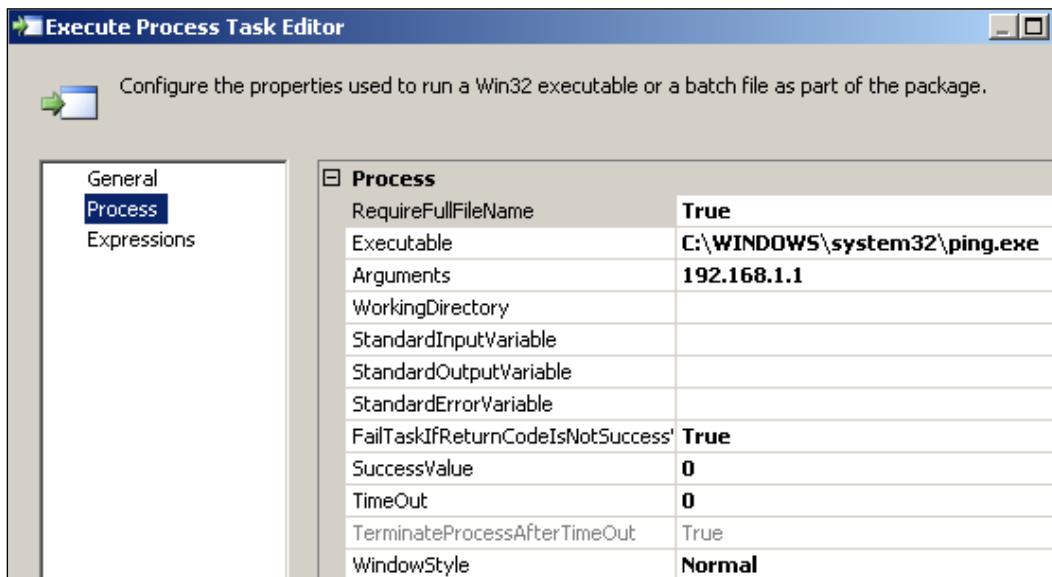
In this task, we would like to ping the default gateway of this network, which happens to be at the IP address, 192.168.1.1. (You may type ipconfig /all at a DOS prompt to get the IP address of your default gateway.) If there is no response, then the Internet is not accessible.

Windows IP Configuration
Ethernet adapter Local Area Connection:

```
Connection-specific DNS Suffix . : 
IP Address . . . . . : 192.168.1.100
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
```

The ping.exe is usually found at the windows directory in Microsoft Windows OS.

4. Modify the entries in the **Execute Process Task Editor**, as shown in the following screenshot. You may have to browse the file system to locate the file path for the executable. Clicking on an empty area along the line item **Executable** would open the dialogue for locating the path.

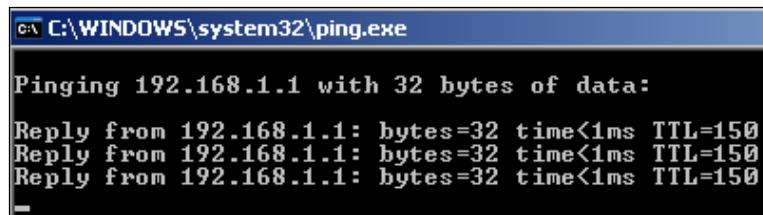


This completes the second event handler for the EvntHnd.dtsx package.

Step 6: Build and Execute the Package with Event Handlers and Verify

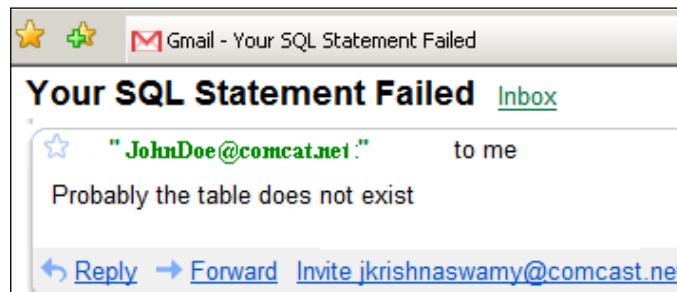
1. Build the Project Ch 16 and execute the package EvntHnd.dtsx.

You will notice a series of events; the EventHnd.dtsx with the **Select from a Table Execute SQL Task** fails (turns red); the Execute Package Task of the EventHnd.dtsx's **OnError** event handler turns green; and the **Ping the Default Gateway Execute Process Task** turns green and finally, the **Failure Message Send Mail Task** turns green sending the email to your email address. When the EventHnd.dtsx gets executed, you will also see DOS screen flashing, showing the success of the ping operation.

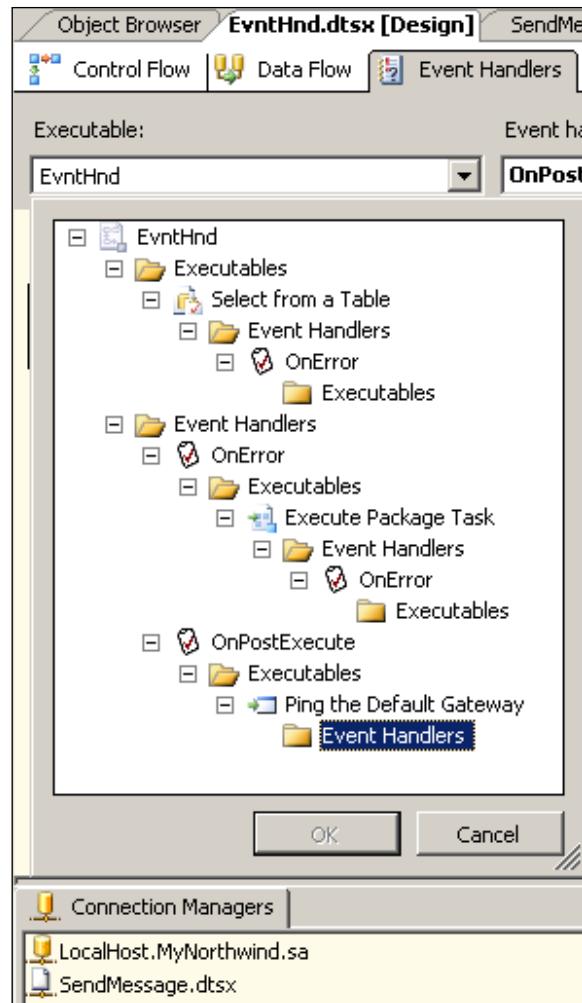


```
C:\WINDOWS\system32\ping.exe
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time<1ms TTL=150
Reply from 192.168.1.1: bytes=32 time<1ms TTL=150
Reply from 192.168.1.1: bytes=32 time<1ms TTL=150
```

Of course, you may also verify to make sure that the email message was sent by checking your inbox (email address removed in the screenshot below).



The Event Handler belongs to a collection of Event Handlers, which may be expanded, as shown in the following screenshot. You may also review in the package explorer page of the Canvas. Also note in the following screenshot that the SendMessage.dtsx is shown in the **Connection Managers**' page of the Canvas, because this package is now nested within the **Execute Package Task** of the "OnError" event handler.



Summary

This chapter described the mechanics of setting up a package in which an event handler is used. The package in the example, raised two events, and two event handlers were configured. A nested package containing a Send Mail Task was executed when the error in the main package was generated. OnPostExecute event was handled by executing an Execute Process Task to verify the network status of the default gateway. Despite the fact it is a contrived example, it illustrates the basic ideas behind event handling in SSIS.

17

Package that Transfers a File Using an FTP Task

FTP short for **File Transfer Protocol** is the best known for what it does – it transfers files in ASCII or other formats, over the Internet, even between disparate operating systems with provisions for file storage. Please review RFC 765 for most of its features. You can also use command line as well as proprietary software programs to download / upload files using FTP.

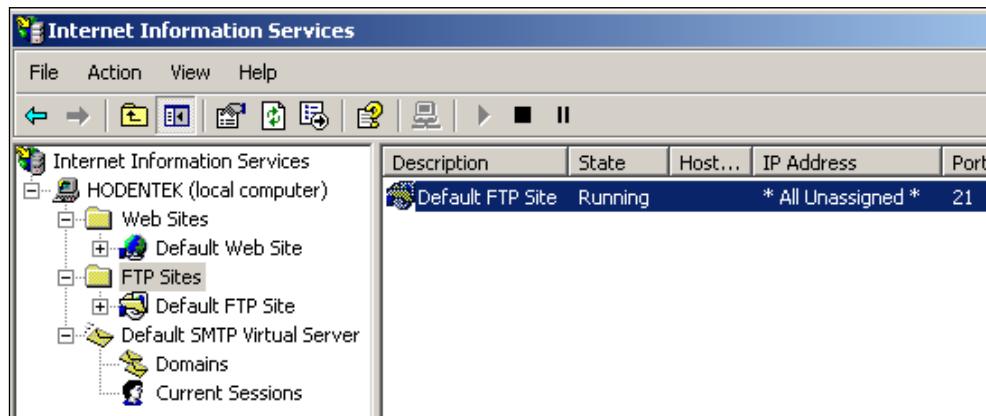
FTP is one of the most common ways in which files (data, text, csv files, documents, music, picture, etc.) are downloaded from websites. It is bidirectional and, therefore, you can upload as easily as download. This computer-to-computer transfer of files obviously will involve authentication. In the anonymous login mode, where the download site allows anonymous logins, you normally do not provide a password while the username is Anonymous. However, for other sites you may need to be authenticated.

FTP task was also a part of the DTS, the **Data Transformation Services** available in SQL 2000. You may review this article, <http://www.aspfree.com/c/a/MS-SQL-Server/Using-Data-Transformation-Services-part-7-Transferring-Files-with-DTS/>, if you want to know how the FTP task was handled in SQL 2000.

In this exercise, you will create a package that incorporates a FTP Task. In the first example, you will be transferring a graphic file (GIF) from a location on your hard drive to the local FTP server. In the second example, you will be downloading a non-ASCII file over the Internet from a site that allows anonymous logins.

Hands-On Exercise: Creating a Package with a FTP Task

In example 1, you will be transferring a text file on the C:\ drive to the root folder of your local FTP server. In IIS 5.1, the version of Internet Information Services program on this machine, the FTP services are also handled by the IIS, as shown in this default view. For the example shown in this exercise to work, you must have the FTP services enabled. This can be enabled through following, Start | Control Panel | Add or Remove Programs. Then, you need to bring up the Add / Remove Windows components. In the Windows Components Wizard, you must enable the Internet Information Services check box.

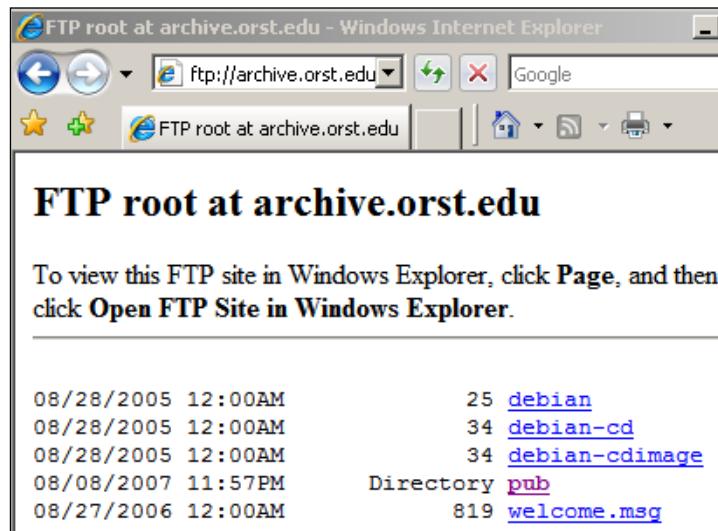


For this server, anonymous authentication has been enabled and users can login without a password. When you browse the site (`ftp://yoursite`), you will be able to see the directory of this site, as shown in the following screenshot. You may also browse using the IP address instead of the computer name. As you can see, this service is not very much utilized on this computer.

The screenshot shows an FTP browser window titled 'Index of ftp://localhost/'. It displays a list of files in the root directory:

Up to higher level directory			
 daruma.gif	20 KB	8:31:00 AM	
 default.htm	1 KB	10:38:00 PM	
 FilesystemTskFile	6 KB	11:03:00 AM	
 POI.txt	4 KB	10:30:00 AM	

In example 2, you will be accessing a site that is public and does not require authentication. We will access the FTP site, `ftp:\\archive.orst.edu`, and download some documents from there.



The following steps will be described in setting up a BI project with an FTP task:

- Create a BI project and rename the default package.
- Build and execute the FTP task.
- Disable the FTP task, add and configure another FTP task to download a file from an Internet public site.
- Build and execute the new FTP task.

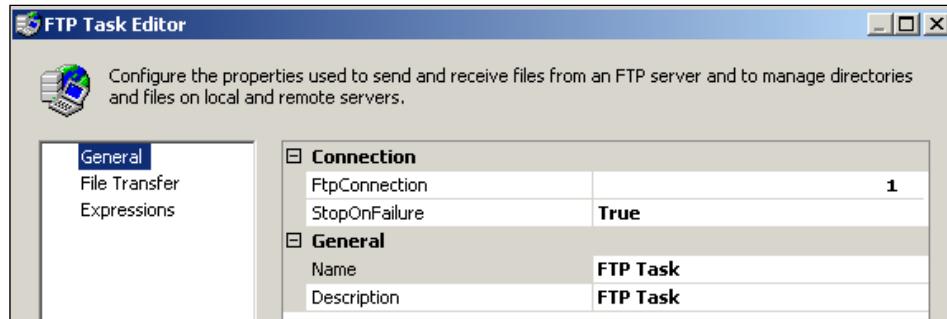
Step 1: Create a BI Project and Rename the Default Package

This process has been described a number of times in various chapters.

1. Create a Business Intelligence Project Ch 17 and rename the default name of the package.
The default name was changed to `ftp.dtsx` in this exercise.
2. Drag and drop an **FTP Task** from the **Control Flow Items** group, from the **Toolbox**, onto the **Control Flow** page of the Canvas.
3. Right-click the **FTP Task**, and from the drop-down choose **Edit...** (or double-click the **FTP Task** component).

Package that Transfers a File Using an FTP Task

This opens the **FTP Task Editor** window, as shown in the following screenshot. (The location shown as **1** is added after the window is open.)

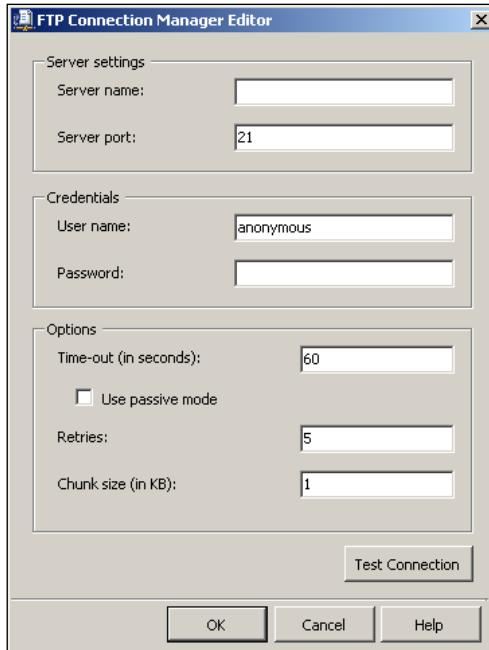


In this window, open with the **General** list item; you need to provide information regarding the **FTP Connection** as well as a **Name** and **Description** of your choice.

In this exercise, the **Name** chosen was "Transfer File Task" and the **Description** was "FTP file to root folder".

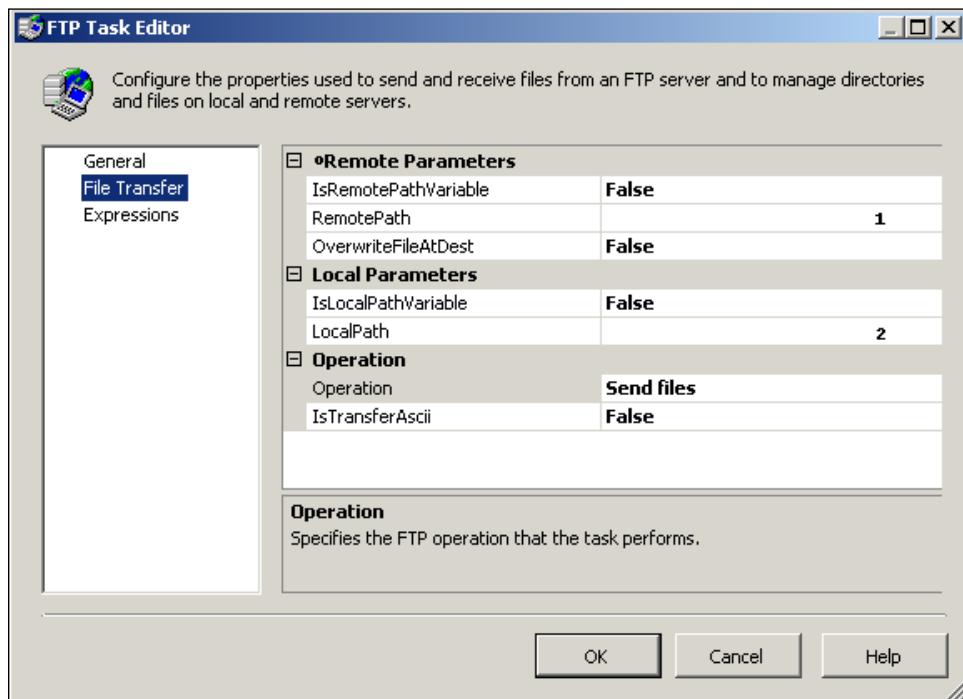
4. Click on the location shown by **1** in the above window along the line item **FtpConnection**, and from the drop-down choose <New connection...>.

This opens up the **FTP Connection Manager Editor** window.



As mentioned earlier, you need to provide a name for the FTP Server in the line item, **Server name**. For the local server, *localhost* will be an appropriate name. This may also be the IP address or the computer name.

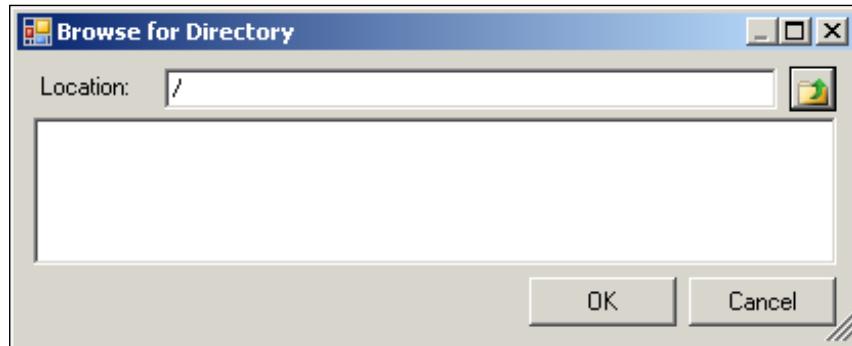
5. Type in **Localhost** and accept all other defaults. As the access is anonymous login, there is no need to provide a password. You may test to verify that the connection is good by clicking on the **Test Connection** button.
6. Click on the **OK** button on the **FTP Connection Manager Editor** window. The name **FTP Connection Manager** gets added to the **General** list item along with the line item, **FtpConnection**, in the **FTP Task Editor** window. Accepting **StopOnFailure** to be *true* will stop FTP task.
7. Click on the **File Transfer** list item, which opens the window (as shown in the screenshot) where you need to provide the information for the remote host and the local computer.



The line items which have boolean values **False** can be set to **true** by clicking on "false" and choosing the drop-down value **true**. The meaning of these items can be read at the bottom, which is now displaying what **Operation** means in this **FTP Task Editor**. Presently, the **Operation** is set to **Send files** option, which means that you are sending a file using this task if this option is chosen.

Package that Transfers a File Using an FTP Task

8. Click on the point indicated by **1** along the line item. **RemotePath** opens the **Browse for Directory** window, as shown in the following screenshot.



This is already at the top level of the local FTP site directory.

9. Accept this choice and click on the **OK** button on this window.

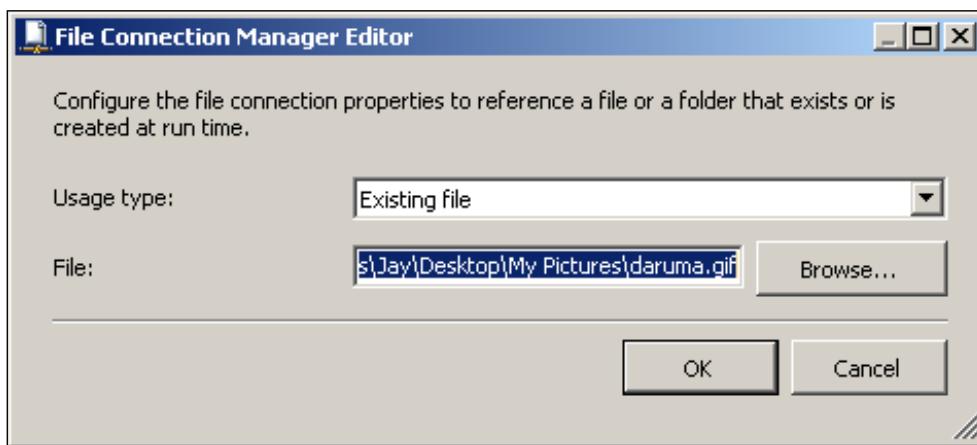
This inserts "/" for the line item **RemotePath** in the **FTP Task Editor**.

10. Now click on the point **2** along the line item **LocalPath** in the **FTP Task Editor** and from the drop-down click on <New connection...>

This brings up the **File Connection Manager Editor**'s window that you have seen in earlier chapters.

11. Using the **Browse** button, look through the hard drive and choose the file (any file of your choice will do) to be transferred.

For this exercise, the graphic file **daruma.gif**, from the hard drive, was chosen, as shown in the following screenshot. The **IsTransferAscii** set to **false** (default) is appropriate.



12. Click on the **OK** button in the above window to complete the **File Transfer Task Editor** items.

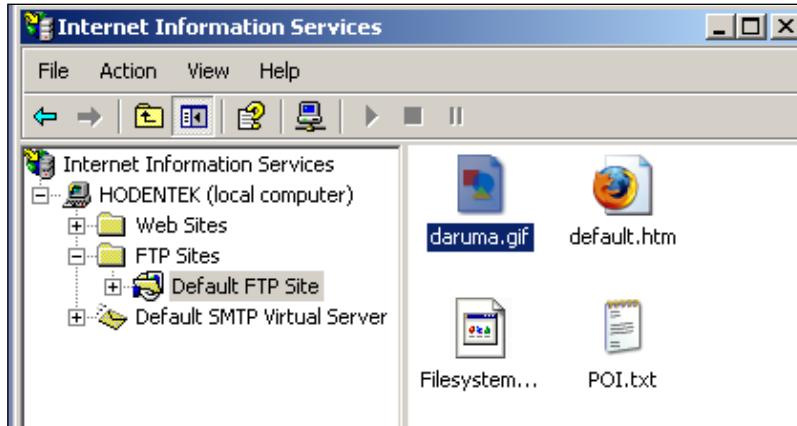
Step 2: Build and Execute the FTP Task

1. Build the project and execute the task or project as done in the previous chapters.

The program begins to run and the **FTP Task** component turns green indicating a successful execution.

2. Open the **Internet Information Services** application and browse the **Default FTP Site**.

You will find that the **daruma.gif** has been transferred (uploaded) to your FTP server.



Step 3: Disable the FTP Task, Add and Configure another FTP Task to Download a File from an Internet Public Site

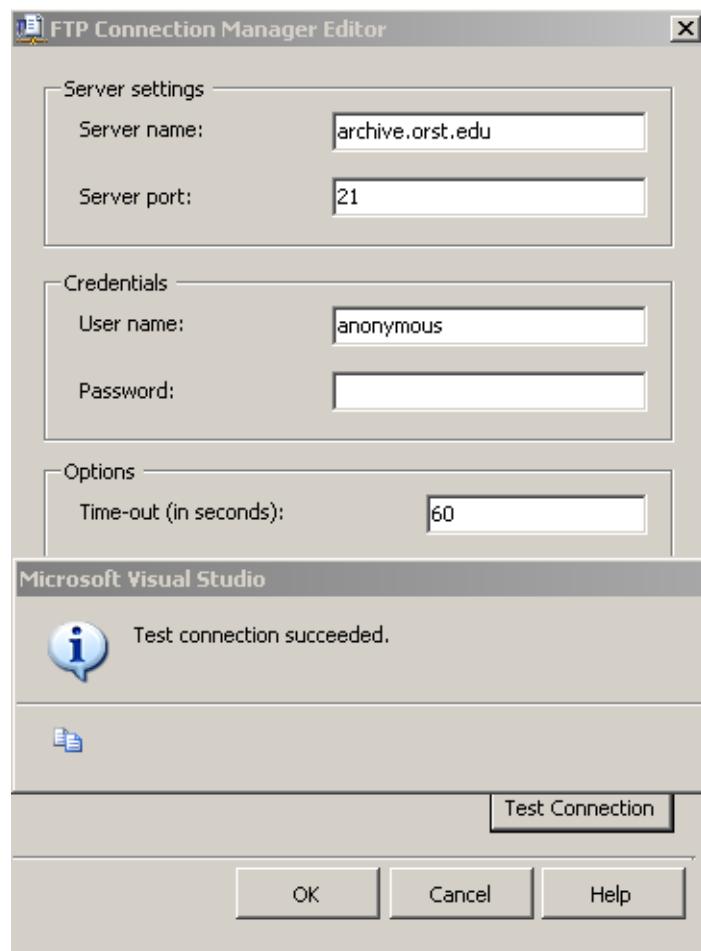
1. Disable the Transfer File Task component on the Canvas by right-clicking the component and choosing **Disable**. It is not really necessary as each task can be executed independent of the other. This is just to keep an eye on the component to be executed.

This will gray out the component.

2. Drag and drop a **FTP Task** component (from the **Control Flow Items** group) from the **Toolbox** onto the **Control Flow** page of the Canvas.

Package that Transfers a File Using an FTP Task

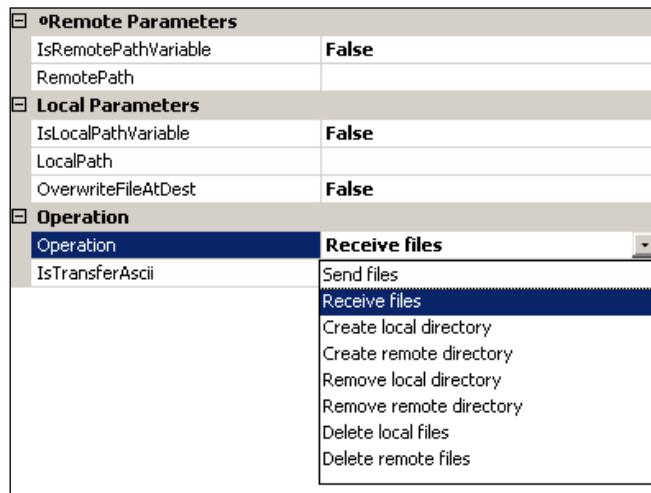
3. Double-click the component and make changes to the **General** list item page of the **FTP Task Editor**. For **FtpConnection** use the connection information as shown in the following screenshot.



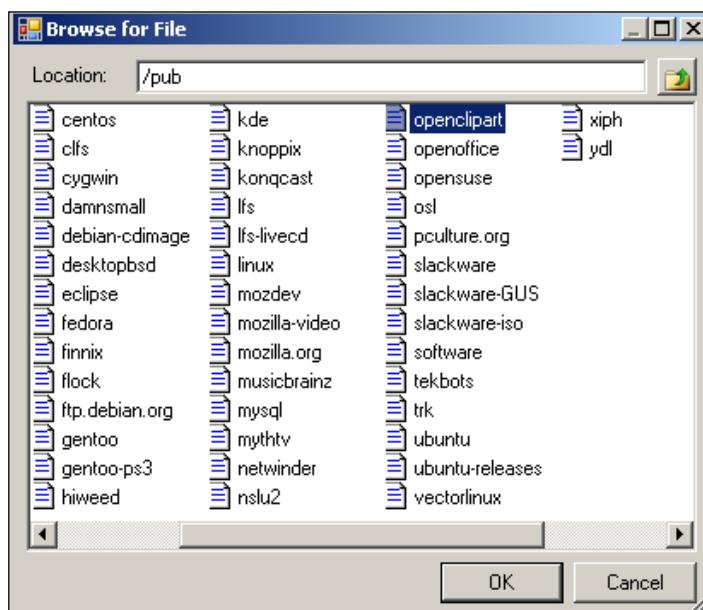
You will be using the host part of the URL for the **Server name**. Details regarding FTP URL format is described in this article: <http://tools.ietf.org/html/draft-casey-url-ftp-00>. You may recall that in the previous task, which was disabled, we used the hierarchical symbol "/" for the top level.

In this example, the **Name** of the task is "Download from FTP site" and the **Description** is "Download a non-ASCII file from Internet". It can be any **Name** and **Description** you choose.

- Click on the **File Transfer** list item. As it is a download operation, change **Send files** (default) to **Receive files** from the drop-down, as shown in the following screenshot.

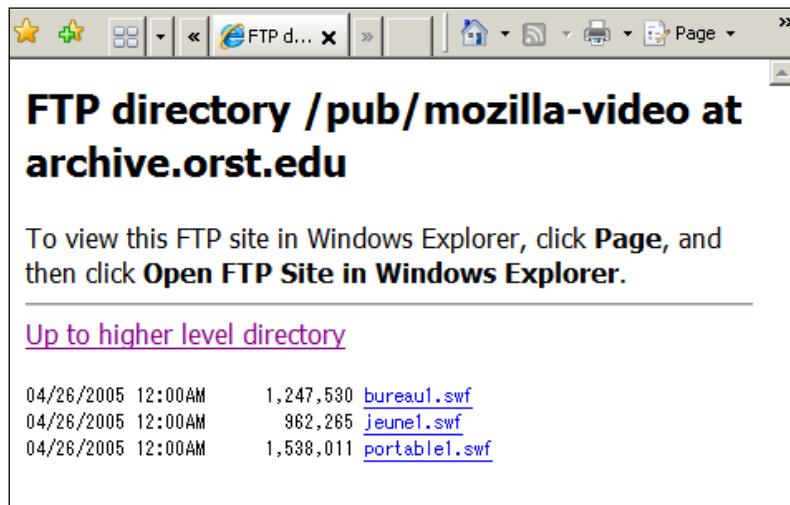


- Follow this by a click along the line item **RemotePath** to browse the site. This opens the **Browse for Directory** window shown earlier.
- Click on the pub folder to go down to the pub directory, and dig further, as shown in the following screenshot.



7. The file in the **mozilla-video** folder in the above directory contains the file we would like to download, as shown in the following screenshot.

Though the FTP Task is intended for file transfer process, the FTP Task Editor does not give access to the file.



8. In the **Browse for file** window, click on **mozilla-video** and click on the **OK** button to close this window.

This will bring you back to the **FTP Task Editor** and the **RemotePath** is now displaying, **/pub/mozilla-video**.

9. Now, modify this line to indicate the file, such as **jeune1.swf** as in **/pub/mozilla-video/jeune1.swf**.
10. Click along the line item **Local Path** and choose to create a folder **OrstEdu** as a repository for the downloaded files, as in the previous example.

This completes the FTP task configuration.

Step 4: Build and Execute the New FTP Task

1. Build the project and execute the newly configured task.

The program runs, and after execution, the file is transferred. You may now look up the **OrstEdu** directory to review the file.

C:\Documents and Settings\Jay\Desktop\OrstEdu			
Name	Size	Type	Date Modified
jeune1.swf	940 KB	Shockwave Fla...	8/21/2007 4:32 PM

Summary

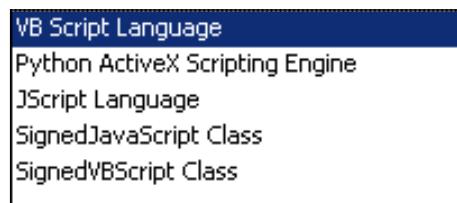
This chapter described the usage of the FTP task. Two examples were considered. In the first example, a graphic file on the hard drive was transferred to the root directory of the FTP site. In the second example, a video file was downloaded to a directory on the hard drive from an Internet public site. The FTP Task Editor GUI correctly identifies the directory structure but did not locate the files. One needs to know the file to download before using the FTP task. It may be noted that the *Remote Path* that the GUI locates depends on the *Operation* you choose for the File Transfer.

18

Package with an ActiveX Script Task

ActiveX was one of the main programming backbones of DTS, the tool that was replaced by SSIS in SQL 2005. The ActiveX Script Task in SSIS is provided for backward compatibility, especially useful for migrating DTS packages that have custom code using a scripting language. Support for ActiveX in SSIS is very limited, and Microsoft recommends using the faster and more stable Script Task using the .NET framework classes. The ActiveX Script Task will be discontinued in the future versions.

The ActiveX Script Task can be created through code and every detail needed for scripting is provided by the Microsoft.SQLServer.DTS.Tasks.ActiveX Script Task namespace. There is no intellisense support, but the task can cater to several scripting languages as shown below.



`CreateObject ()` method is a much used native method for creating objects (Automation Objects). The syntax used for creating an object is::

```
CreateObject (servername.typename [, location])
```

Most of the Microsoft Office applications' automation objects are created using this method. For example, the following VBScript statements create objects in various applications:

```
Set WordApp=CreateObject ("Word.Application", "Hodentek") would create  
a MS Word application on the computer "Hodentek"
```

```
Set xlObj=CreateObject ("Excel.Application") would create an  
MS Excel application.
```

Once the Application Object is created, then one has to know the Application's Programming Interface (API) to access methods and properties. Objects can also be created using the above syntax for Internet Explorer, SQL Data Management Objects, Data Transformation Services, ADODB, FileSystemObject, etc. The ActiveX Script Task creates these objects and works with them. Although, the examples shown in this chapter all use VBScript (default), the other languages shown in the previous screenshot can also be used.

Hands-On Exercise: Creating a Package with ActiveX Script Tasks

In this chapter, we will be working with three examples. In the first example, you will be using the ActiveX Script Task to access a word document to which you write a short sentence (a sentence with some errors), and then you will be using the properties and methods of the Word Object Model to count the number of spelling errors.

In the second example, you will be using an ActiveX Script Task to create a SQL Data Management Object to determine the number of tables in the database of a named server and the name of a table given its ordinal number.

In the third example, you will be connecting to Internet Explorer to bring up the web page of the author's blog at: <http://hodentek.blogspot.com>.

All tasks will be created in the same package as individual tasks can be executed independently.

Example One: Word Automation

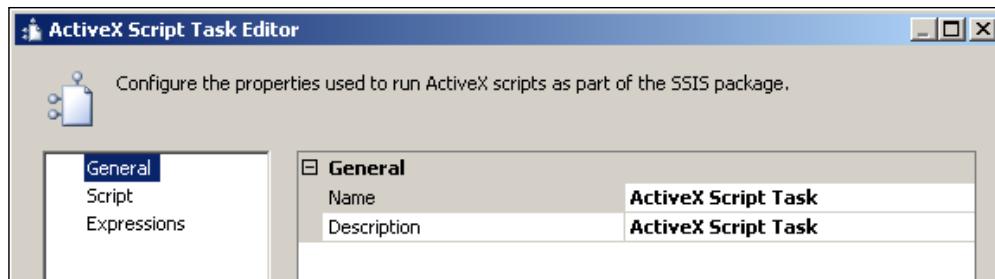
Step 1: Creating a BI Project and Adding an ActiveX Script Task to the Package

1. Create a Business Intelligence Project Ch 18 and rename the default name of the package.

The default name was changed to ActiveXDemo.dtsx in this exercise.

2. Drag and drop an **ActiveX Script Task** from the **Control Flow Items** group from the **Toolbox** onto the **Control Flow Page** of the Canvas.
3. Right-click the **ActiveX Script Task** and from drop-down choose **Edit...** (or double-click the **ActiveX Script Task** component).

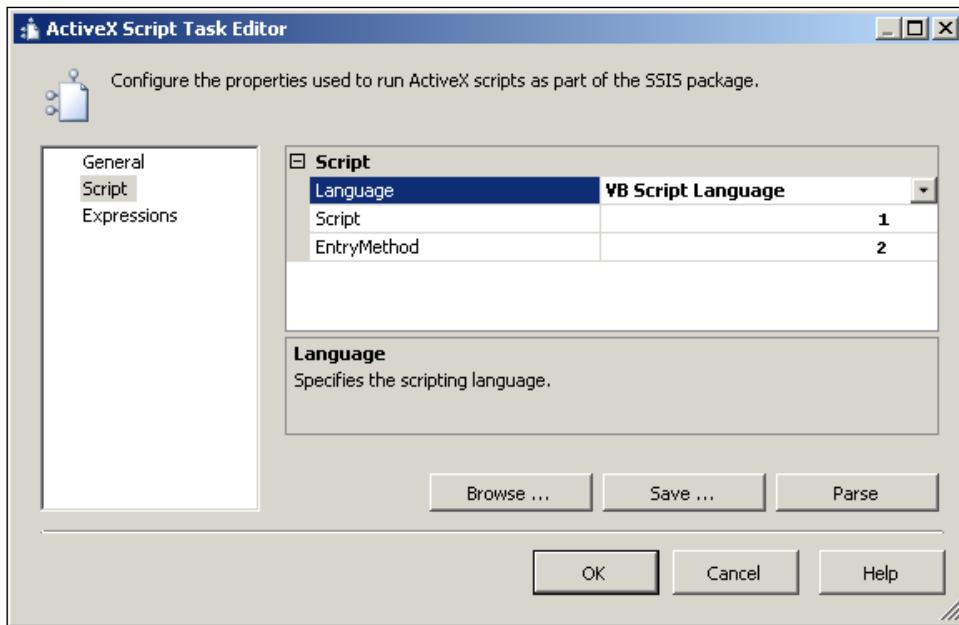
This opens up the **ActiveX Script Task Editor**, as shown in the next screenshot. This window is same for all the tasks we will be creating. You can provide a **Name** and a **Description** for this task.



In this example, the **Name** and **Description** are as follows, respectively, "Count Spelling Errors" and "Count Number of errors in a Microsoft Word Document".

4. Make the above changes to **Name** and **Description** line items and click on the **Script** list item on the left.

This opens the **ActiveX Script Task Editor's** *Script* related page, as shown in the following screenshot with the default, **VBScript Language**.



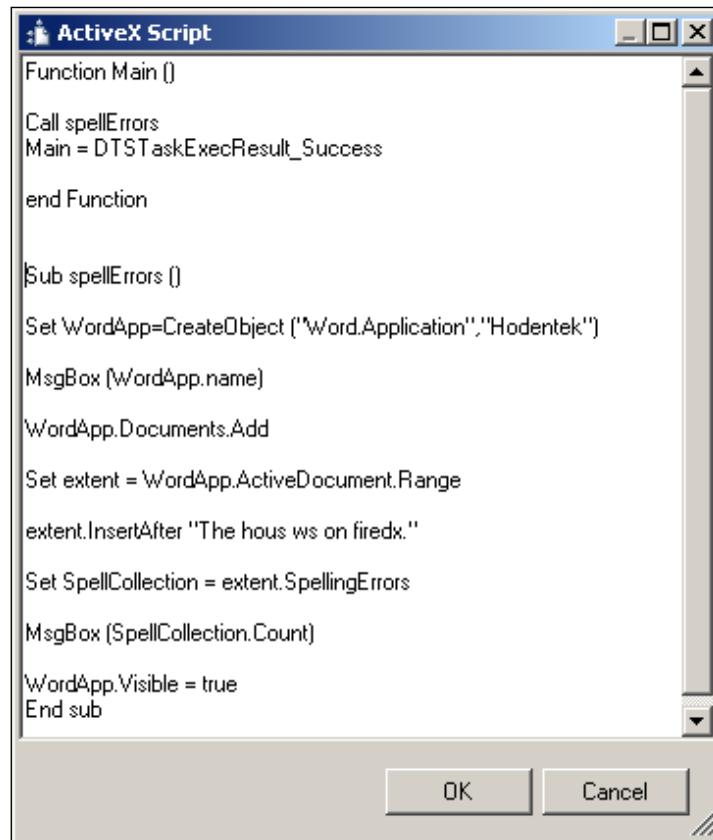
5. Click near the point marked **1** along the line item **Script**, as shown in the previous screenshot.

This opens a text editor window, *ActiveX Script*, which is initially empty. The developer is supposed to type in the code.

It is also possible to look for a saved script file (*.txt) on the machine using the **Browse...** button. This window also allows you to *parse* the script or *save* the script using appropriate buttons.

In the present example, we will be using the following script to create an instance of Microsoft Word Automation object, and then present it with a sentence with errors, and programmatically find the number of spelling errors present in the sentence. The ActiveX Script editor gets displayed and it is initially empty.

6. Type in the text into the **ActiveX Script** text Editor as shown in the following screenshot.



The following is the code with comments:

```
Function Main ()  
'This next statement calls the sub procedure  
Call spellErrors  
Main = DTSTaskExecResult_Success  
End Function  
  
Sub spellErrors ()  
'Create a word automation object with the next stataement  
Set WordApp=CreateObject ("Word.Application", "Hodentek")  
'Display the programmatic name of the application object  
MsgBox (WordApp.name)  
'Add a document to the Documents collection
```

```
WordApp.Documents.Add
'Establish a range in the active document
Set extent = WordApp.ActiveDocument.Range
'Insert a short sentence with errors using the insertAfter method
extent.InsertAfter "The hous ws on firedx."
'Use SpellCollection to collect errors in the sentence
Set SpellCollection = extent.SpellingErrors
'Display the number of errors in a message box.
MsgBox (SpellCollection.Count)
'Display the word document
WordApp.Visible = true
End sub
```

7. Click on the point indicated by **2** and type in **Main**.

This is the entry point for the script and the code begins to execute from this point.

8. Build the project and execute the **ActiveX Script Task—Count Spelling Errors**.

The first message box shows **Microsoft Word** as the name of the application. The second window shows the number 3, 3 being the number of errors in this short sentence. Finally, the word document pops-up containing the short sentence.

Example Two: Finding the Number of Tables in a Database

Step 2: Adding Another ActiveX Script Task to the Project

1. Disable the **ActiveX Script Task—Count Spelling Errors** by right-clicking the component and choosing **Disable**.
2. Drag and drop an **ActiveX Script Task** on to the Canvas from the **Control Flow Items** group in the **Toolbox**.

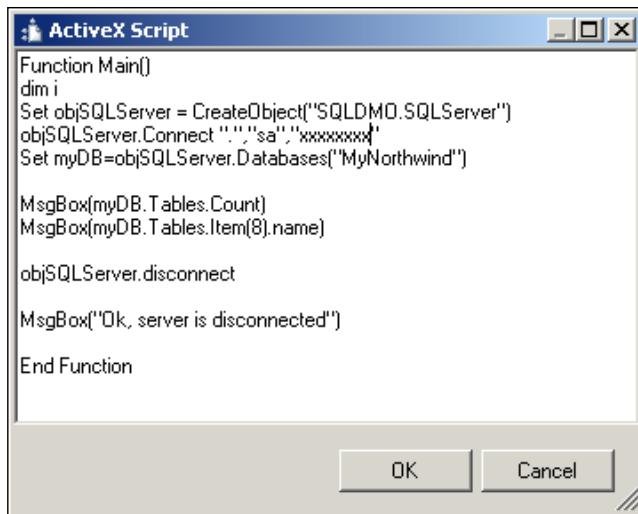
This brings up the **ActiveX Script Task Editor** window, as described previously.

3. In the **General** list item, provide a **Name** and a **Description** for the component.

For this example, the **Name** is "**How many Tables?**", and the **Description** is "**Finding the number of tables in a database**".

4. In the **Script** list item, click and invoke the **ActiveX Script** text editor window, as shown in the previous example.
5. Type in the following script in the **ActiveX Script** text editor window.

The window should appear as shown in the next screenshot.



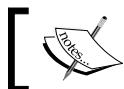
Fully commented code is given below.

```

Function Main ()
    'The function Main is the entry point
    Dim i
    'The next statement creates a SQL DMO object
    Set objSQLServer = CreateObject ("SQLDMO.SQLServer")
    'Connecting to the SQL Server, provide Server name,
        Username, Password
    objSQLServer.Connect ".", "sa", "xxxxxxxx"
    'Provide a reference to the database on this server.
    Set myDB= objSQLServer.Databases ("MyNorthwind")
    'Count the number of tables in this database and display it
    MsgBox (myDB.Tables.Count)

    'Find the name of the table with the ordinal number 8 and display it
    MsgBox (myDB.Tables.Item (8).name)
    'Disconnect the object from the server
    objSQLServer.disconnect
    MsgBox ("Ok, server is disconnected")
End Function

```



Note: The password will be visible in this script.



Using the SQL DMO Object model many other tasks can be accomplished.

6. In the line item **Entry method**, type in **Main** as before.
7. Build the project and execute the **ActiveX Script Task** component in the 'Canvas' – "**How many Tables?**".

The program runs and the first message box shows 57, the number of tables in this database, and the next message box shows "Poem" – the name of the table whose ordinal number is 8. The final message displays that the DMO object was disconnected from the server. You can verify the above by executing the stored procedure `sys.sp_help` in the SQL Server 2005 Management Studio, as shown in the following screenshot.

The screenshot shows a SQL Server Management Studio window titled '(local).MyNort...SQLQuery1.sql*'. The query entered is 'exec sys.sp_help'. The 'Results' tab is selected, displaying a table of database objects. The table has columns: #, Name, Owner, and Object_type. The data shows 10 user tables, with the 8th row ('Poem') highlighted. The table structure includes columns for User_type, Storage_type, Length, and Prec at the bottom.

#	Name	Owner	Object_type
1	Customers	dbo	user table
2	CustSsis	dbo	user table
3	DataConvert	dbo	user table
4	Employees	dbo	user table
5	OracleView	dbo	user table
6	Order Details	dbo	user table
7	Orders	dbo	user table
8	Poem	dbo	user table
9	Prblm	dbo	user table
10	Products	dbo	user table

Example Three: Navigating to the Internet Explorer Browser

Step 3: Adding another ActiveX Script Task to the Project

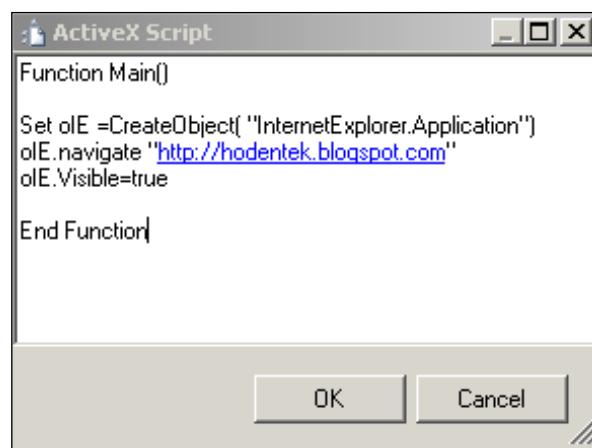
1. Disable the **ActiveX Script Task – How many Tables?**, by right-clicking the component and choosing **Disable**.
2. Drag and drop an **ActiveX Script Task** on to the Canvas from the **Control Flow Items** group in the **Toolbox**.
3. This brings up the **ActiveX Script Task Editor** window, as described previously.
4. In the **General** list item, provide a **Name** and a **Description** for the component.

For this example, the **Name** is "Display IE Browser" and the **Description** is "Navigate to the Internet Explorer Browser".

This brings up the ActiveX Script Task Editor window.

1. In the **Script** list item, click and invoke the Activex Script text editor window, as in the previously example.
2. Type in the following script in the **ActiveX Script** text editor window.

The window should appear as shown in the following screenshot. Of course you can type in the URL of your own liking.



There are really three lines in the script. The first line creates the object, the second line navigates to the site whose URL is given as an argument to the method, and the third line displays the page.

3. In the line item **Entry method** type in **Main** as before.
4. Build the project and execute the **ActiveX Script Task** component in the Canvas – "Display IE Browser".

The program cranks up and the **ActiveX Script Task** component turns green indicating success and the IE browser gets displayed.

Summary

Three different ActiveX Scripts were demonstrated in this chapter. For demonstration purposes, message boxes were chosen and this is not recommended in an actual task as it is counter productive. Also not recommended, is the password visibility in clear text. ActiveX Script Task is not going to be supported in future versions and it is better to take heed of Microsoft's recommendation to use the Script Task described in the next chapter.

19

Package with a Script Task

There is an ocean of difference between the ActiveX Script Task and the Script Task in SSIS. Script Task has all the .NET Framework classes that can be easily accessed through the Object Browser and of course, the intellisense support that is invaluable, two of the many things that contribute to increased productivity. ActiveX Script Task, as we saw in the previous chapter, has none of these. It is no wonder that ActiveX Script Task is poorly described and the recommendation has always been to replace it with the Script Task. ActiveX Script Task is a task that will fade away in the next version of SSIS.

VSA (Visual Studio for Applications), the replacement for the well known VBA, is the scripting engine that drives the Script Task in SSIS. VSA was designed to add extensibility to the program, so Script Task adds extensibility to the SSIS programming. It's ideal for tasks that cannot be performed by the out-of-the-box tasks (which are already numerous) that SSIS provides. Sometimes, it can also be used for combining the functionality of several tasks without using their graphic counterparts. It is generally understood that a script is interpreted, but the SSIS Script Task scripts are compiled and executed when the package is executed. The language supported by the Script Task is presently limited to Microsoft Visual Basic .NET (probably more developers use VB.NET as opposed to C#).

Overview of the Hands-On Exercises

In this chapter, we will be working with the following simple examples:

- **Simple Calculation**

Given two numbers calculate their sum using the Script Task.

- **Calculation using variables**

In the previous example, the summands were assigned in the code, but in this example they are defined variables in the scope of the script.

- **Add an imports statement to build a string**

In this example, the `System.Text` class is added to build up strings using the script.

- **Retrieve data from a database table in the SQL Server 2005**

In this example, you will connect to the SQL Server 2005 and retrieve data from a table on one of its databases.

- **Combine the last two examples in displaying data and copying to a file**

In this example, the data retrieved is displayed using a string builder and also written to a new text file created on-the-fly.

Hands-On Exercise: Creating a Package with Script Tasks

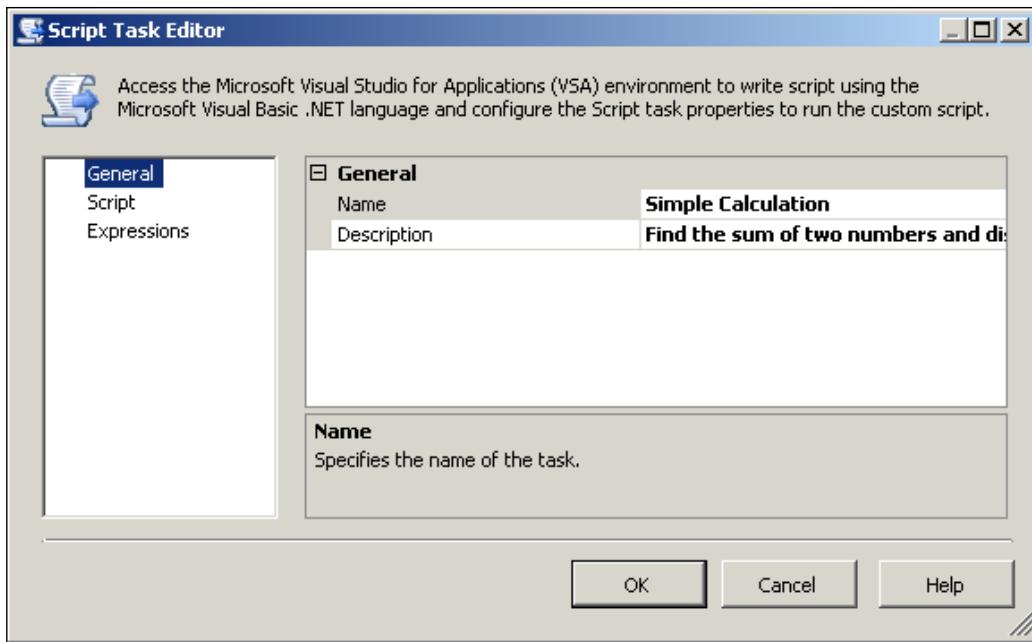
In this chapter, we will just create one project and test the various examples using individual Script Tasks in the 'Canvas'. It is not necessary to disable the task to execute the other task(s), as it is possible to just execute the task you need to execute.

1. Create a Business Intelligence Project, **Ch 19**, and rename the default name of the package.
The default name was changed to `scripts.dtsx` in this exercise.
2. Drag and drop a **Script Task** from the **Control Flow Items** group from the **Toolbox** onto the **Control Flow** page of the Canvas.
3. Right-click the **Script Task** and from the drop-down choose **Edit...** (or double-click the **Script Task** component).

Simple Calculation

This opens up the **Script Task Editor**, as shown in the next screenshot. This window is common to all tasks we are going to test in this package. You can provide a **Name** and a **Description** for this task.

1. Provide a **Name** and **Description** for the task, as shown in the screenshot, by typing them in the editor.



2. Click on the **Script** list item on the left of the **Script Task Editor**.

This will open the **Script** item details on the right-hand side of the **Script Task Editor**, as shown in the next screenshot.

There are five Script-related line items:

- **Script Language**

Well, there is only one Script Language, Microsoft VB.NET.

- **PrecompileScriptIntoBinaryCode**

The default is *true* and in all the examples in this chapter it is set to *false* either in this **Script Task Editor** or the **Properties** window of this Task Component. With this set to *False*, the code will be compiled just before the package (task) is executed. With the other option set to *true*, the code will be compiled and stored as a Base64 code in the package.

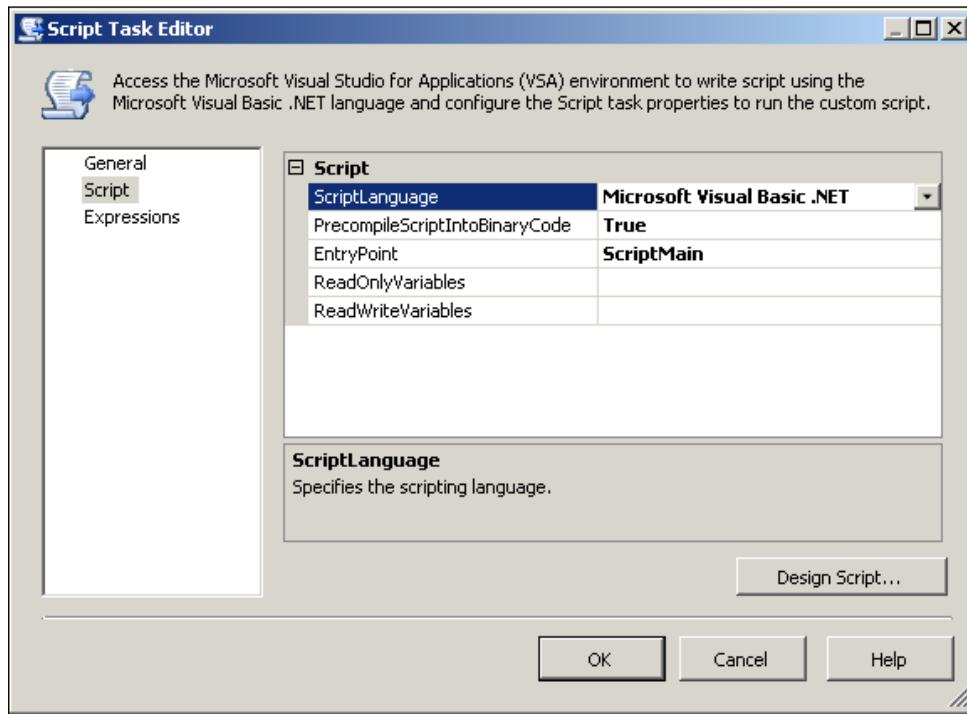
- **Entry Point**

This is where the script loads into the engine. **ScriptMain** is the name of the VB Class file.

- **ReadOnlyVariables**

Package with a Script Task

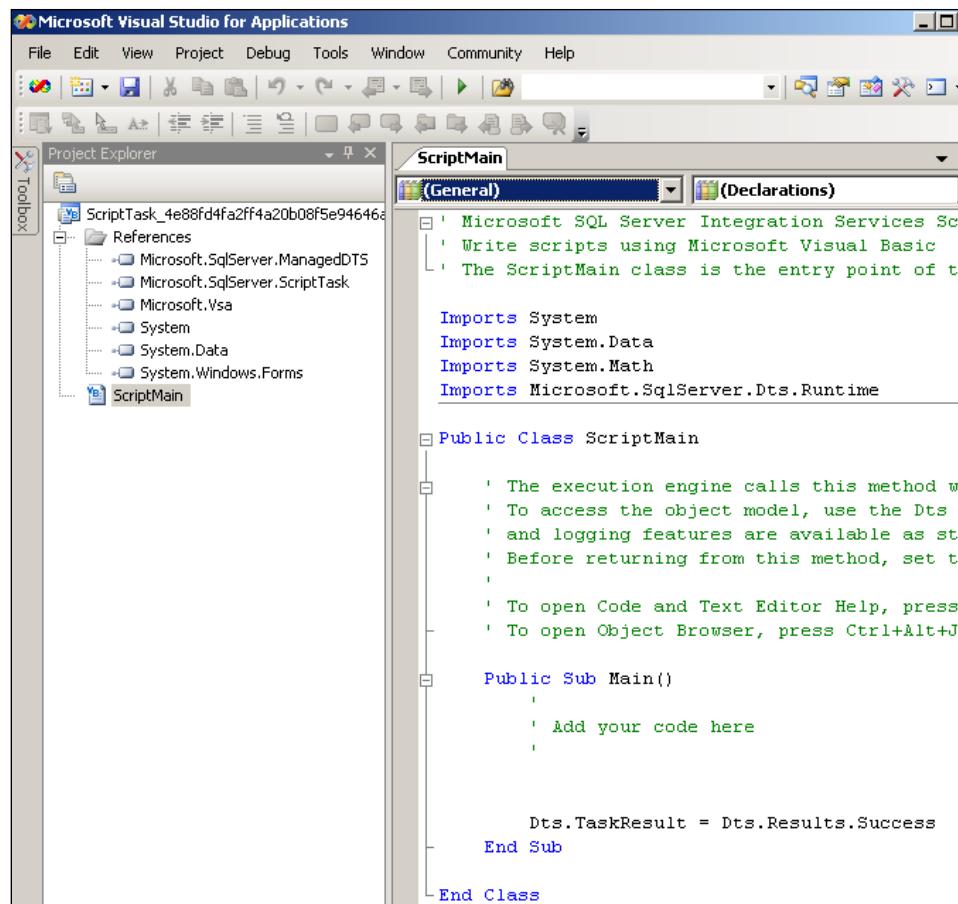
- The variable(s) are *read only* as the line item suggests.
- **ReadWriteVariables**
- The variable(s) are *read/write* as the line item suggests.



The button at the bottom, **Design Script...** opens a new Script editor interface where most of the code is created.

3. Change the **PrecompileScriptIntoBinaryCode** line item to **false** and click on the **Design Script...** button.

This opens the **Microsoft Visual Studio for Applications** window as shown in the next screenshot with some template code. It shows up, referencing a number of libraries as seen in the **References** node shown expanded in the screenshot. The **Imports** statements for the **Public Class ScriptMain** are also in this template code. You need to write your code at the position "**Add your code here**".



4. For this task, just type in the code shown in the next paragraph to replace the block.

```

Public Sub Main()
    '
    ' Add your code here
    '

    Dim a As Integer
    Dim b As Integer
    a = 5
    b = 10
    MsgBox (a * b)
    Dts.TaskResult = Dts.Results.Success
End Sub

```

5. Click on the **OK** button on the **Script Task Editor** window.

For this example, we do not need to go to the **Expressions** line item.

6. Build the project and execute the "**Simple Calculation**" **Script Task**.

You should immediately see 50 in a pop-up **Script Task** message window.

Calculation using variables

1. Drag and drop a **Script Task** from the **Control Flow Items** group from the **Toolbox** on to the **Control Flow Page** of the 'Canvas'.
2. Right-click the **Script Task** and from the drop-down choose **Edit...** (or double-click the **Script Task** component).

This opens up the **Script Task Editor**, as shown in the next screenshot. This window is common to all tasks we are going to test in this package. You can provide a **Name** and a **Description** for this task.

3. Provide a **Name** and **Description** for the task by typing them in the editor. The **Name** is, "**Calculate Using Variables**". The **Description** is, "**Get variables, calculate and Display result**".
4. Right-click on an empty area in the **Control Flow** page of the 'Canvas' and from the pop-menu choose **Variables**.

This brings up the **Variables** window that is empty to start with. Use the **Add Variable** icon twice to add two variables **a**, and **b**, of type **Int32**. You can also specify what value they should have. This has been described in detail in an earlier chapter.

Name	Scope	Data Type	Value
a	scripts	Int32	25
Variable1	scripts	Int32	10

5. In the half-finished window, type over **Variable1** with **b**.

The variables **a**, and **b** are now defined in the **Scope** of the "scripts" and available for the scripts.

6. Change the **PrecompileScriptIntoBinaryCode** line item to **false** in the **Script** list item of the **Script Task Editor**, and click on the **Design Script...** button.

7. Add the two variables as shown in the next screenshot to the Script Task Editor in the line item, **ReadOnlyVariables**.

Script	
ScriptLanguage	Microsoft Visual Basic .NET
PrecompileScriptIntoBinaryCode	False
EntryPoint	ScriptMain
ReadOnlyVariables	a,b
ReadWriteVariables	

8. Click on the button **Design Script...** to open the Microsoft Visual Studio for Applications window and type in the following code at the indicated location.

```
Public Sub Main()
    '
    ' Add your code here
    Dim x As Variable
    x = Dts.Variables(0)
    Dim y As Variable
    y = Dts.Variables(1)
    Dim c As Integer
    MsgBox(CInt(x.Value) * CInt(y.Value))

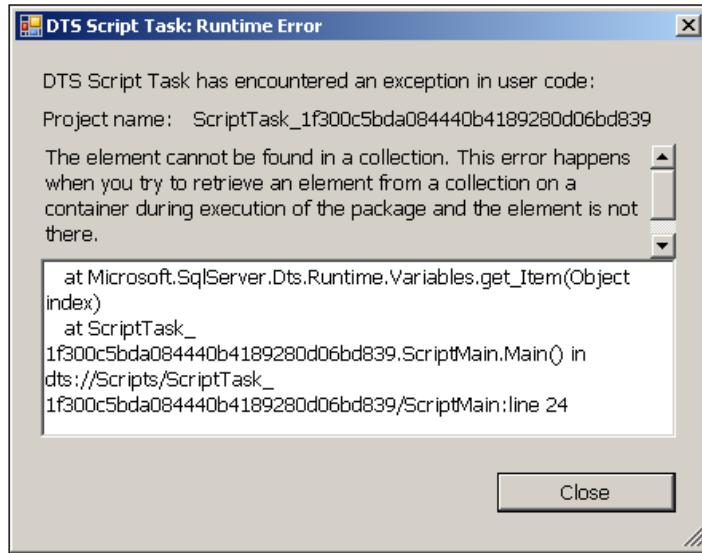
    Dts.TaskResult = Dts.Results.Success
End Sub
```

It is easy to see that **x** is referencing the variable **a** given by `Dts.Variables(0)`. The variables collection's first element is **a** and its value is set to 25. Similarly, the value of **y** is set to 10 through the variable **b**. Since **c** is of type Integer, the variables are changed to Integer by casting.

9. Click on the **OK** button on the **Script Task Editor**.
10. Build the project and execute the "**Calculate Using Variables**" Script Task. The program cranks up, the script Component turns green and soon you will see a pop-up window showing 250.

What if you don't include the variables a, and b in the Script Task Editor?

Well, when you try to execute the **Script Task** you will see the following message delivered by the **DTS Script Task Runtime Error**, which in this case appears to be quite meaningful.



Add an Imports Statement to Build a String

1. Repeat the previous steps to configure a new Script Task whose **Name** is "**String Builder Task**", having a **Description** "**Build a string of numbers**".
2. Change the **PrecompileScriptIntoBinaryCode** line item to **false** in the **Script** list item of the **Script Task Editor**, and click on the **Design Script...** button
3. In the **Microsoft Visual Studio for Applications**, type in the following code in the appropriate location as shown in the following paragraph.

```
Public Sub Main()  
    '  
    'Add your code here  
    '  
    Dim i As Integer  
    Dim sb As New StringBuilder("", 200)  
    For i = 0 To 15  
        If i = 15 Then  
            sb.Append(i)  
        Else  
            sb.Append(i)  
            sb.Append(" | ")
```

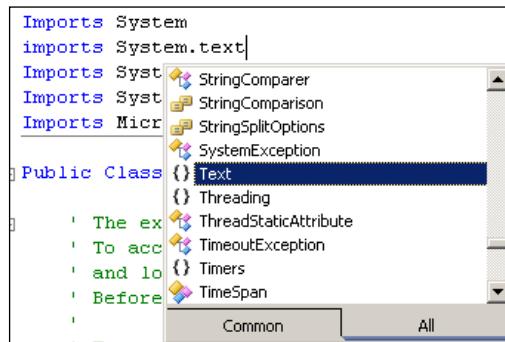
```

        End If
    Next
    MsgBox(sb.ToString)
    Dts.TaskResult = Dts.Results.Success
End Sub

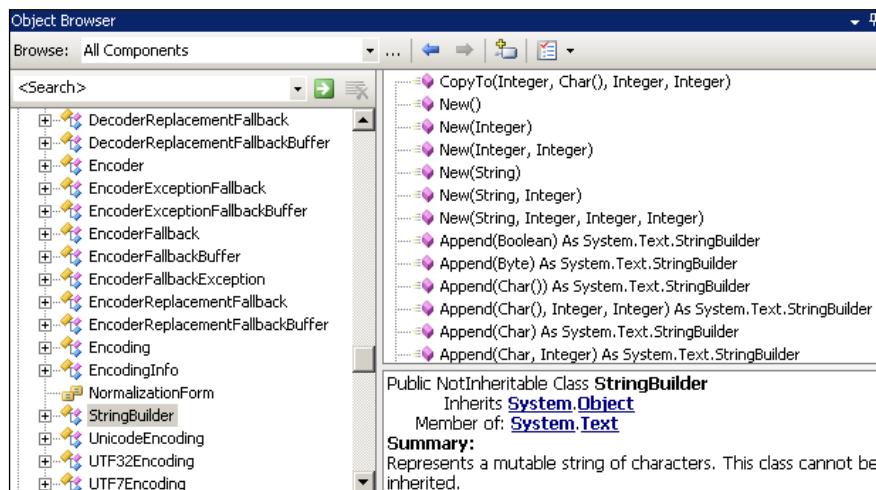
```

The code is very simple. The *For...Next* loop steps through 0 to 15 and included in the loop is a string builder variable *sb*, which adds a number followed by a " | " symbol. The *If...Else...End...If* statement takes care of removing " | " from the end.

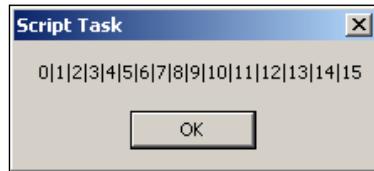
In order to use the **StringBuilder**, a class that helps in building strings, we have added an extra imports statement (*imports System.Text*) to the default imports statement that you find in the **Microsoft Visual Studio Editor's ScriptMain** as shown in the next figure *a la* Intellisense.



This will give access to the methods and properties of this class, as shown in the object browser.



4. After adding the code as described, click on the **OK** button on the **Script Task Editor** window to close the window.
5. Build the project and execute the "**String Builder Task**" Script Task.
6. The program runs and the Script Task component turns green and soon you will see the **Script Task** message window pop-up, as shown.



Retrieve Data from a Database Table in the SQL Server 2005

1. Repeat the previous steps to configure a new Script Task whose Name is "**Get Employees Task**" with the description "**Get last names from MyNorthwind database's Employees table**".
2. Change the **PrecompileScriptIntoBinaryCode** line item to **false** in the **Script** list item of the **Script Task Editor**, and click on the **Design Script...** button.
3. In the **Microsoft Visual Studio for Applications**, type in the following code in the appropriate location, as shown in the following paragraph. Make sure to add a line continuation character in the second statement if the whole statement is not in one line.

```
Public Sub Main()
    '
    ' Add your code here
    '

    Dim con As New SqlClient.SqlConnection
    con.ConnectionString = "Data Source=.;User ID=sa;
    password=xxxxxxxx;Initial Catalog=MyNorthwind;
    Persist Security Info=True;"
    con.Open()

    Dim cmd As New SqlClient.SqlCommand
    cmd.Connection = con
    cmd.CommandText = "Select LastName from Employees"
    Dim rst As SqlClient.SqlDataReader
    rst = cmd.ExecuteReader
```

```
Dim i As Integer
i = rst.FieldCount
MsgBox(i)

Dim j As Integer
While rst.Read
    MsgBox(rst.Item(j))
End While
Dts.TaskResult = Dts.Results.Success

End Sub
```

The above code is basically quite simple. The steps are as follows:

- Declare a connection object that uses the SQLClient native provider.
- Assign the string to this connection. This may be obtained by configuring a DataReader Source component, and copying the string from the Connection Manager's property window.
- Open the connection.
- Declare a **SQL Command** object.
- Assign the above connection to the Command's connection.
- Provide the SQL for the Command Text of the above Command. This is the statement that retrieves the Last Names from the employees table in the *MyNorthwind* database on the local SQL Server 2005 declared in the Connection String.
- Declare a **SQL DataReader** object.
- Assign the proceeds of the Command's ExecuteReader method to the SQL DataReader.
- In the While...End, While block pulls out the employees' Last Name one-by-one and displays them in a Script Task window.
- 4. Click on the **OK** button in the **Script Task Editor** window to close the window.
- 5. Build and execute the Script Task, "**Get Employees Task**".

The program starts running and soon you will see the Last Name of the employees from the employee's table displayed in the Script Task message window. The first message displays the number of columns retrieved (which is 1 in this example). The second and subsequent message boxes that pop up display the employees' names one-by-one as the variable *j* is cycled in the loop.

Combine the Last Two Examples in Displaying Data and Copying to a File

1. Repeat the previous steps to configure a new Script Task whose **Name** is "Copy Employees names to a file" having a **Description** "Get last names from MyNorthwind database's Employees table and write to a text file".
2. Change the **PrecompileScriptIntoBinaryCode** line item to **false** in the **Script** list item of the **Script Task Editor**, and click on the **Design Script...** button.
3. In the **Microsoft Visual Studio for Applications**, type in the following code in the appropriate location, as shown in the following paragraph.

```
Public Sub Main()
    '
    ' Add your code here
    '

    Dim con As New SqlConnection
    con.ConnectionString = "Data Source=.;User ID=sa;
    password=venugopal;Initial Catalog=MyNorthwind;
    Persist Security Info=True;"
    con.Open()

    Dim cmd As New SqlCommand
    cmd.Connection = con
    cmd.CommandText = "Select LastName from Employees"
    Dim rst As SqlDataReader
    rst = cmd.ExecuteReader

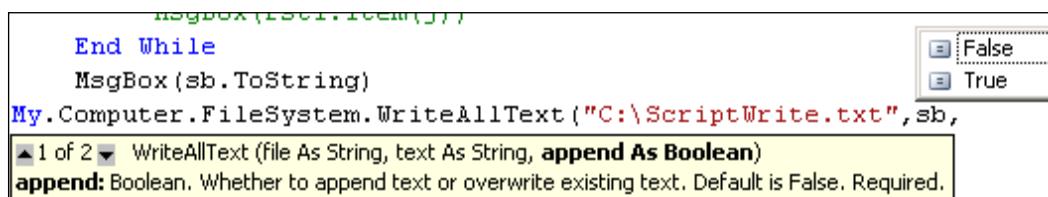
    Dim i As Integer
    i = rst.FieldCount
    MsgBox(i)

    Dim sb As New StringBuilder("", 200)
    Dim k As Integer
    While rst.Read
        sb.Append(rst.Item(k))
        sb.Append(vbCrLf)
        'MsgBox(rst1.Item(j))
    End While
    MsgBox(sb.ToString)
    My.Computer.FileSystem.WriteAllText(
        "C:\ScriptWrite.txt", sb.ToString, False)

    Dts.TaskResult = Dts.Results.Success
End Sub
```

As you can see, the `StringBuilder` is brought in to build a string consisting of employee's name, and the final message box displays this string.

In the next statement, you see how the string is written to a text file created on-the-fly. Of course, you get the intellisense help, as shown in the next screenshot.



The screenshot shows the Microsoft Script Task Editor. A code snippet is displayed:

```

    MsgBox(sb.ToString())
My.Computer.FileSystem.WriteAllText("C:\ScriptWrite.txt", sb,
    1 of 2 ▾ WriteAllText (file As String, text As String, append As Boolean)
append: Boolean. Whether to append text or overwrite existing text. Default is False. Required.

```

A tooltip for the `append` parameter is open, showing two options: `False` and `True`.

4. Change the `PrecompileScriptIntoBinaryCode` line item to `false`, in the **Script** list item of the **Script Task Editor**.
5. Click on the **OK** button on the Script Task Editor.
6. Build the project and execute the Script Task "**Copy Employees names to a file**".

The program starts up and the Script Task turns green and you will see a file created in the `C:\` drive, as shown below. The first message box shows the number of fields which is 1 in this example. The second message box shows the names of all the employees in a message box. The `C:\ScriptWrite.txt` file is shown in the next figure.



Summary

This chapter described several examples of using the Script Task. The Script Task has the full power of the .NET framework and is very powerful. It can combine several tasks by few lines of code very easily increasing the productivity. For example, the last script example would have required a Data Flow task with DataReader Source and its connection manager, a File System Task, and a file system connection manager, a precedence constraint, etc. While Script Task can do a lot of heavy lifting, its interface to COM objects is not straightforward as SSIS does not support COM for this task, a real impediment to productivity. The interface for the Script Task is not completely described in this chapter, with emphasis placed on hands-on activity; the reader must acquaint himself/herself with the complete details of the Microsoft Visual Studio application to derive the full benefit.

20

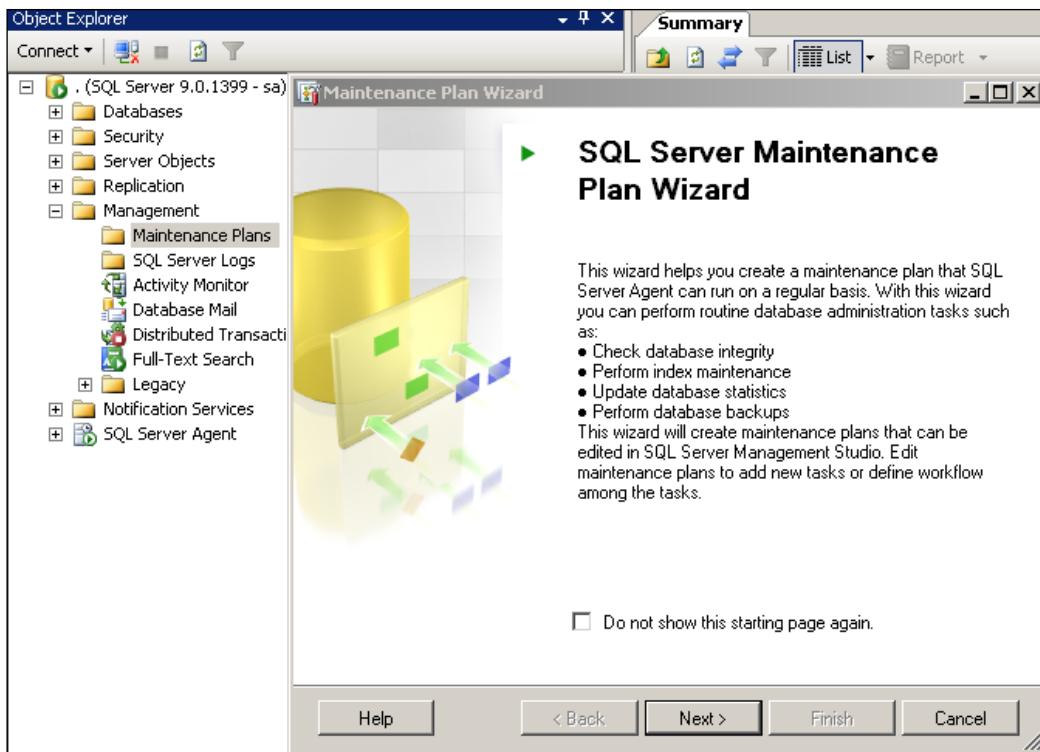
Package with Maintenance Plan Tasks

Database maintenance tasks have been designed to take care of tasks that are needed to maintain a database by keeping it up-to-date and consist of many activities that are needed to be performed either on demand or follow a set schedule. Three items have to be considered for database maintenance:

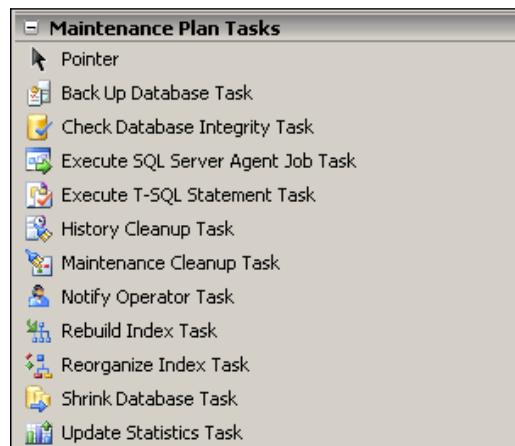
- Optimizations
- Integrity
- Backups

These tasks can also be managed from the **Microsoft SQL Server Management Studio**, as seen using the wizard by making a right-click on the **Management Plans** node in the Management Studio. As you can see, the wizard can be configured to carry out several tasks listed on its opening page. These same tasks will be carried out by SSIS, except that the task is saved by default to the file system. The default where these are stored starting from the wizard will be on the SQL Server. These tasks on the SQL Server are not automatically visible nor can they be invoked in SSIS.

Package with Maintenance Plan Tasks



The Maintenance Plan Tasks group in SSIS has the same tasks you find in the above wizard (as shown in the next screenshot) except that you have an additional task, the **Notify Operator Task**. Notifying the operator in the Management Studio is carried out by the **SQL Server Agent** through the *Alerts* node.



Backing Up a Database

Data loss to business is unacceptable. A copy of the most recent data should always be available. A backup copy of data is therefore indispensable for restoring the database, if for some reason the production data becomes unavailable due to media problems, hardware failures, etc. While backup activity is always paired with restore activity, SSIS has only the backup task. The backup activity requires a lot of planning as to how frequently the data has to be backed-up, what is to be backed-up, data or data files; what type of media will be used for keeping the backup, what type of backup is needed, etc. The **Back Up Database Task** takes these into consideration, as will be seen in the hands-on exercise.

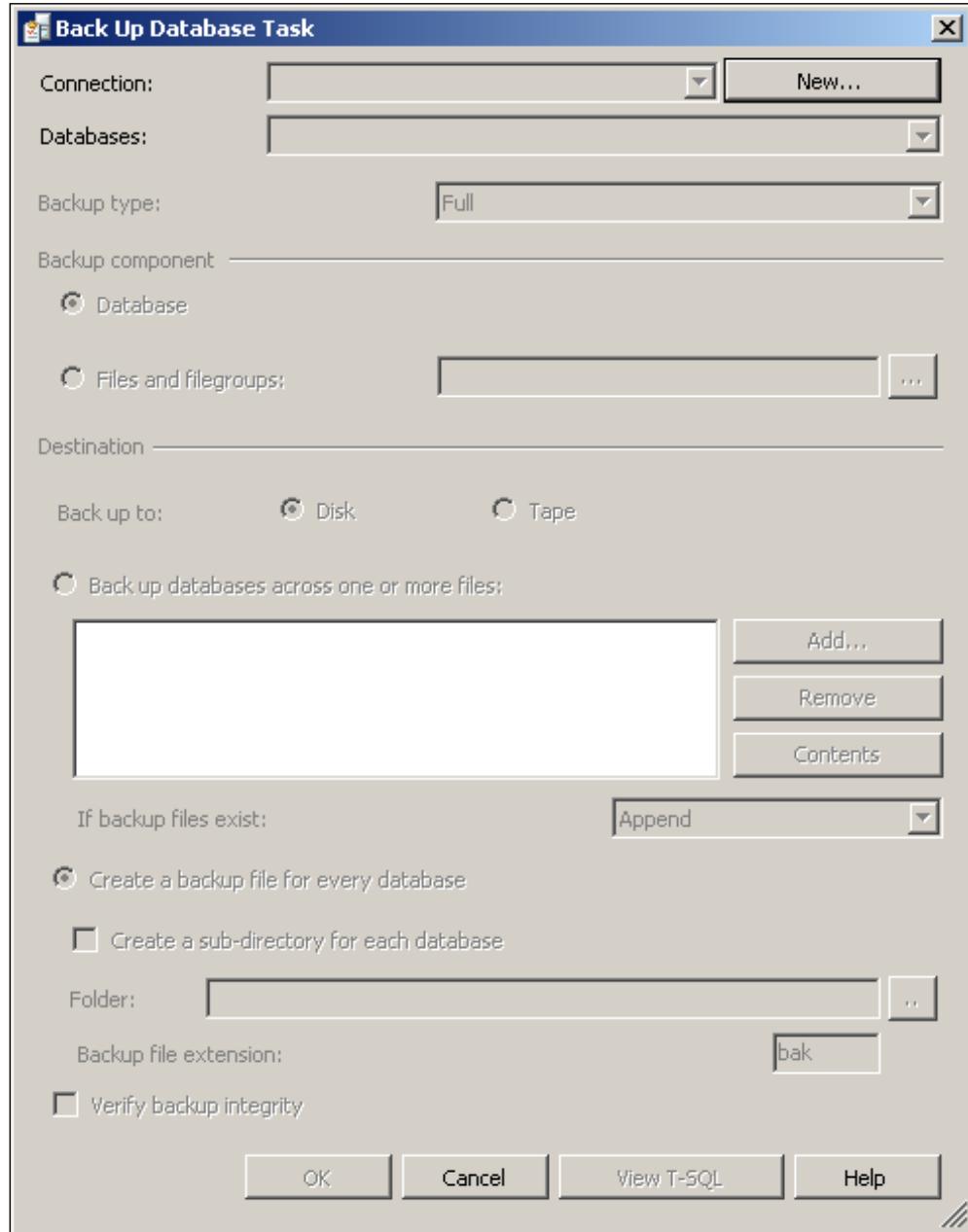
In this chapter, we will just consider one of the options available in the **Back Up Database Task**. The other tasks are as easy to configure.

Hands-On Exercise: Creating a Package with Maintenance Tasks

1. Create a Business Intelligence Project **Ch 20** and rename the default name of the package.
The default name was changed to `Maintain.dtsx` in this exercise.
2. Drag and drop a **Back Up Database Task** from the **Maintenance Plans Tasks** group, from the **Toolbox** onto the **Control Flow** page of the Canvas.
3. Right-click the **Back Up Database Task** and from the drop-down choose **Edit...** (or double-click the **Back Up Database Task** component).

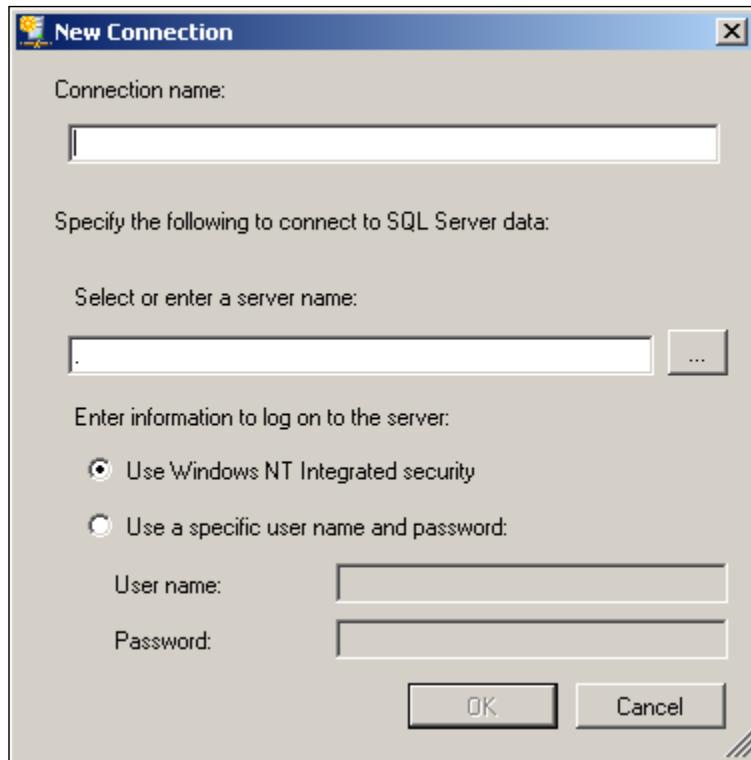
Package with Maintenance Plan Tasks

This brings up the **Back Up Database Task** editor window.



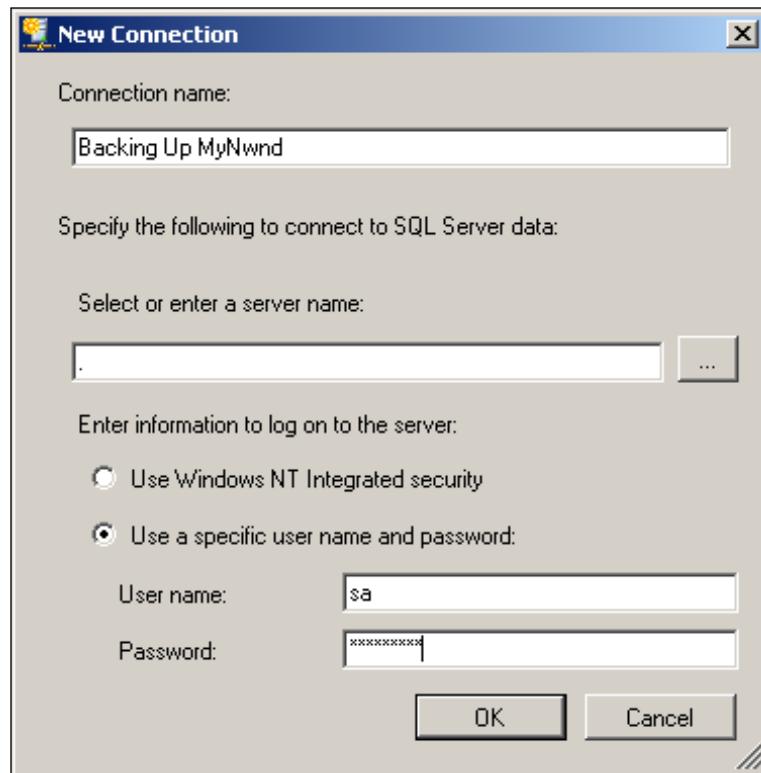
4. Click on the **New...** button along the label, **Connection**.

This opens the **New Connection** window, as shown below, displaying the local server by **"."** in the text box below the label, **Select or enter a server name**.



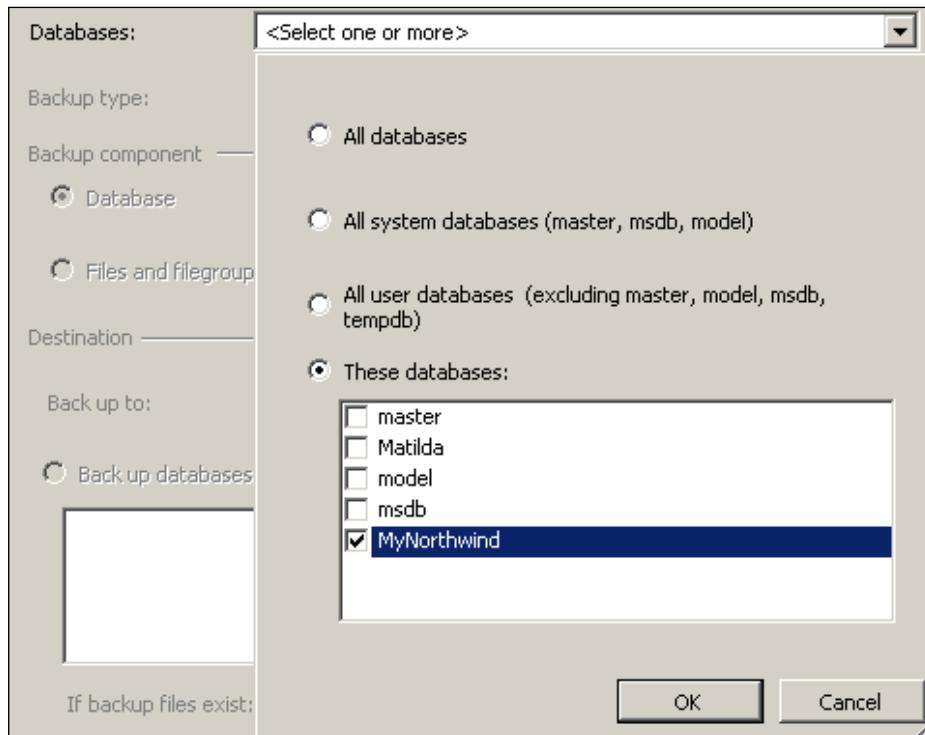
5. Type in a name for the connection.

Here, it is named "**Backing Up MyNwnd**" by typing this into the text box below the label **Connection name**. The authentication will be by SQL Server authentication. Enter the details, as shown, in the next screen.



This enables the **Databases** drop-down list box control, displaying **<Select one or more>**.

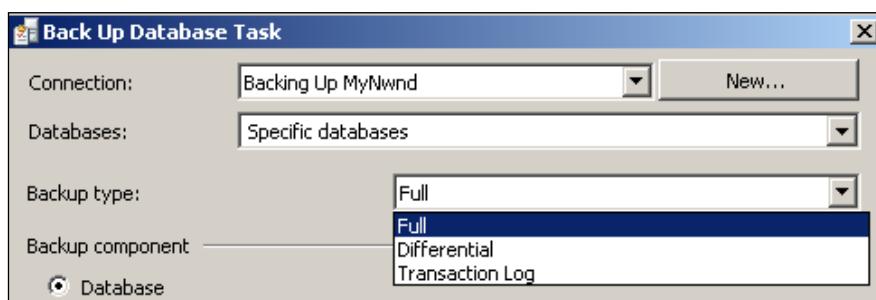
6. Click on this drop-down and select **MyNorthwind**, as shown in the following screenshot.



Of course, you can choose the other databases as well if needed.

7. Click on the **OK** button after making the choice.

The Databases entry now displays **Specific Databases** and the backup type can be chosen by clicking on the **Backup Type** drop-down, as shown in the following screenshot.

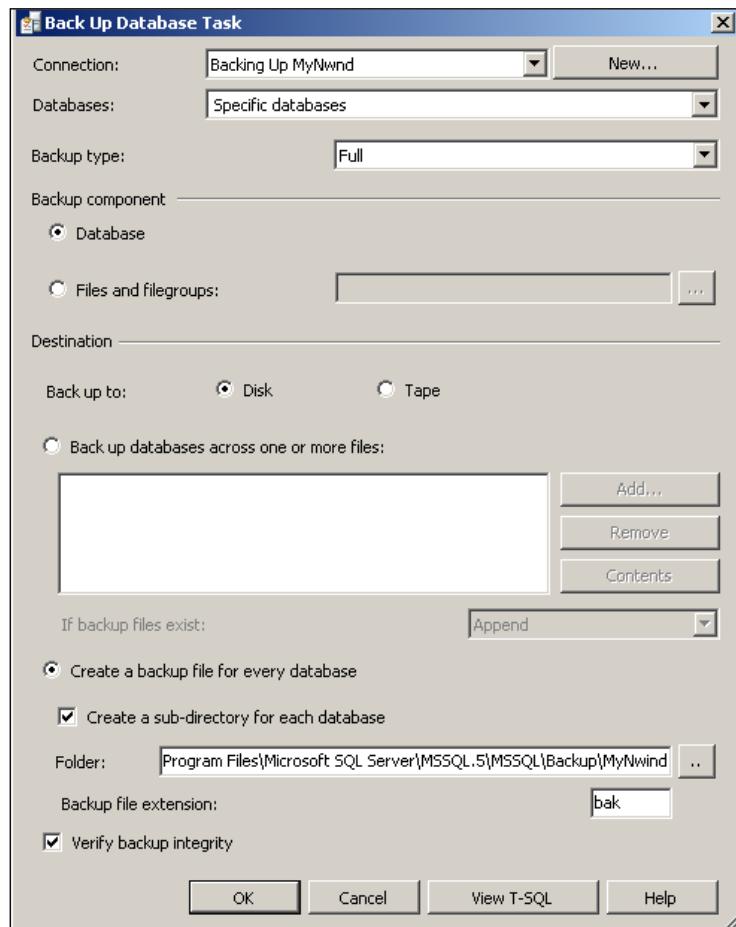


Here, a full backup of the database will be made. Notice that instead of the database, the files can also be backed up by choosing the appropriate option. Differential backups and Transactional log backups are also possible.

Now you need to specify the **Destination**. There are two choices here, the **Disk** or the **Tape**.

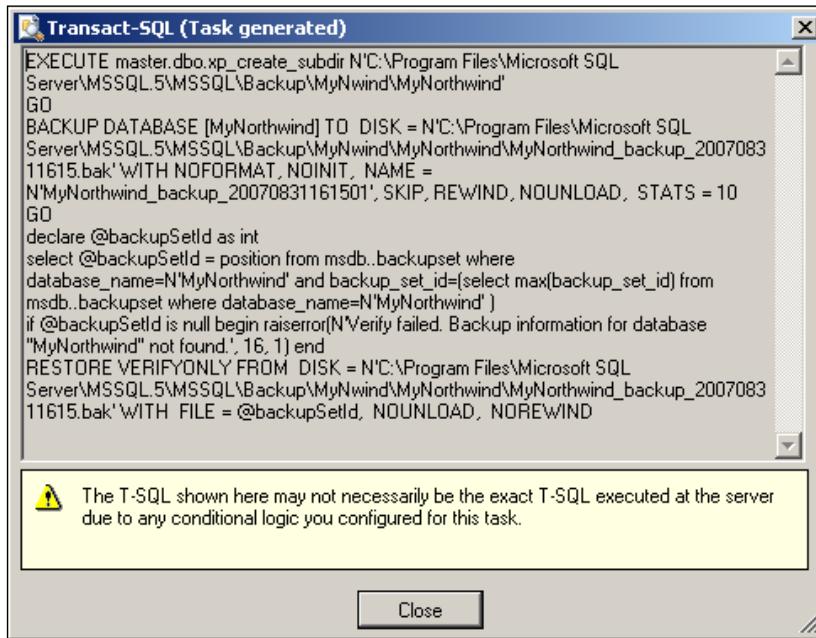
8. In this example, accept the default **Disk**.
9. Accept the default **Create a backup file for every database** option.
10. Create a sub-directory **MyNwind** by placing a check mark in the check box, as shown in the following screenshot.
11. Modify the **Folder** location using the ellipsis button to reflect the above name of the sub-directory.

When the choices made are completed, the **Back Up Database Task** window should be as shown in the following screenshot.



12. Click on the **View T-SQL** button.

This opens the **Transact SQL** pop-up window generated as shown in the next window. This is the T-SQL generated by the task. This has been a straightforward configuration and should work the same on the SQL Server 2005.



```

EXECUTE master.dbo.xp_create_subdir N'C:\Program Files\Microsoft SQL
Server\MSSQL.5\MSSQL\Backup\MyNwind\MyNorthwind'
GO
BACKUP DATABASE [MyNorthwind] TO DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL.5\MSSQL\Backup\MyNwind\MyNorthwind\MyNorthwind_2007083
11615.bak' WITH NOFORMAT, NOINIT, NAME =
N'MyNorthwind_backup_20070831161501', SKIP, REWIND, NOUNLOAD, STATS = 10
GO
declare @backupSetId as int
select @backupSetId = position from msdb..backupset where
database_name=N'MyNorthwind' and backup_set_id=(select max(backup_set_id) from
msdb..backupset where database_name=N'MyNorthwind')
if @backupSetId is null begin raiserror(N'Verify failed. Backup information for database
"MyNorthwind" not found.', 16, 1) end
RESTORE VERIFYONLY FROM DISK = N'C:\Program Files\Microsoft SQL
Server\MSSQL.5\MSSQL\Backup\MyNwind\MyNorthwind\MyNorthwind_2007083
11615.bak' WITH FILE = @backupSetId, NOUNLOAD, NOREWIND

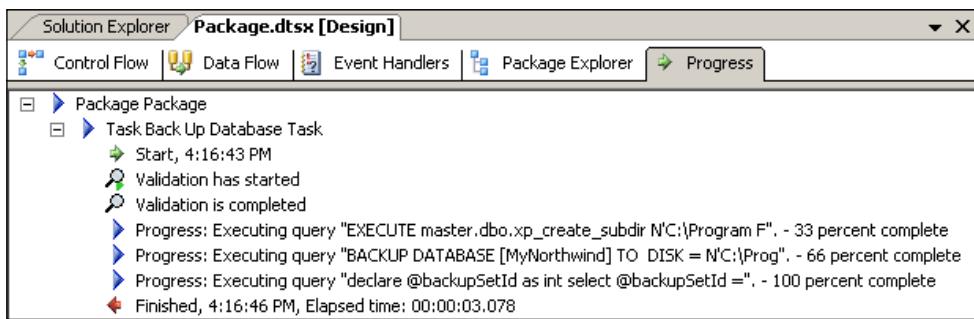
```

The T-SQL shown here may not necessarily be the exact T-SQL executed at the server due to any conditional logic you configured for this task.

Close

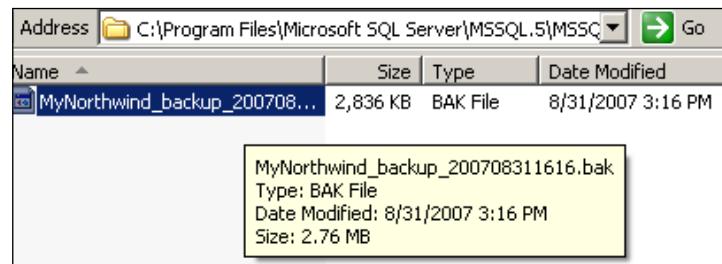
13. Execute the task as in the previous examples by making a right-click with your mouse on the **Back Up Database Task** component in the Canvas.

The task runs, and after a little while the **Back Up Database** component named **Backing Up MyNwnd** in the Canvas turns green. The **Progress** tab of the Canvas shows how the task was executed, as shown in the following screenshot. This includes the extended stored procedure as well as the BACKUP DATABASE statements appropriately.



14. Browse to the location where you created the backup for review.

You will see a new file `MyNwind.bak` in the file location chosen as shown below.



The backup file shows the name of the database as well as the date and time when it was made.

"`MyNorthwind_backup_20078311616.bak`".

Summary

This chapter considered the Maintenance Plan tasks in general and in particular, one of the options that you have in the Back Up Database Task. As mentioned, you have another option in the SQL Server 2005 Management Studio for Maintenance Plan tasks, if you have no access to the Business Intelligence via SSIS. The configuration of the Maintenance Plan Tasks in SSIS is easy and quite straightforward and follows similar wizard driven dialog windows.

Index

A

ActiveX Script task, SSIS
about 263, 264
adding 265-268
BI project, creating 265-268
new ActiveX Script task, adding to project 268-272
package, creating 264

Aggregate Data Transformation
about 147
adding 149
BI project, creating 149
configuring 150, 153
data flow task, adding 149
DataReader source, adding 149
DataReader source, configuring 149
DataReader source, connecting 150
package, executing 156, 157
path, establishing 150
Percentage Sampling data flow item, configuring 154, 155
Percentage Sampling data transformation, adding 153
Percentage Sampling data transformation, path establishing to 153
project, building 156, 157
recordset destination data flow component, adding 155
recordset destination data flow component, configuring 155, 156
results, reviewing 156, 157
steps 149
used 148

B

BI project, SSIS
creating, for integration services 28-32
creating, Visual Studio 2005 used 28
Folders 67, 68
items in Options 66
items in Tools 66
Microsoft products used 27
package, executing 68
package design, canvas 37
project, saving 68
project window, overview 33-36
properties window 32
Property Pages window 67, 68
resources used 28
toolbox 44
Visual Studio 2005, launching 28-32
windows 53

Bulk Insert task
about 123
adding 126
BI project, creating 126
configuring 126-131
data transferring, from flat file to SQL Server database table 123
error, handling 132
flat text file, creating 124
flat text file, using 124
package, executing 131
project, building 131
steps 124
table, creating 124, 125

Business Intelligence. *See BI project, SSIS;*

C

- Character Map Data transformation** 102
- Conditional Split Data Transformation**
 - about 135
 - adding 137
 - BI project, creating 136
 - configuring 137-139
 - data flow task, adding 136
 - DataReader source, adding 136
 - DataReader source, configuring 136
 - DataReader source, connecting 137
 - data transferring, from flat file to SQL Server database table 135, 136
 - package, executing 145
 - path, establishing 137
 - project, building 145
 - recordset destination, adding 140
 - recordset destination, configuring 141-144
 - results, reviewing 145
 - steps 135, 136

D

- Data Conversion Data Flow transformation**
 - about 159
 - Access Destination 160
 - adding 165, 166
 - BI project, creating 162-165
 - configuring 167, 168
 - DataReader source 160
 - data flow task, adding 162-165
 - data transferring, to Excel file 162
 - Excel source 160
 - Excel source, configuring 162-165
 - OLE DB data destination 160, 161, 168
 - package, testing 173, 174
 - path establishing, from Excel source 166, 167
 - project, building 173, 174
 - recordset destination, adding for displaying errors 172, 173
 - recordset destination, configuring for displaying errors 172, 173
 - steps 162
- data flow components, SSIS**
 - data flow destinations 21, 22

data source components 12-14
data transformation 14, 16

data transformation

about 14, 15
contributing, to package design 16-21

Data Transformation Service 7

Diff, XMLTask type

BI project, creating 179
control flow task, adding 179
documents 177
exercise 179
XMLTask, configuring 180-184

DTS 7, 251

E

ETL 7

event handler, SSIS

about 239
BI project, creating 240, 241
default package, renaming 240, 241
Execute Package task, adding to OnError event 244, 245
Execute Process task, adding to OnPostExecute event 246
Execute SQL task, adding 242-244
Execute SQL task, configuring 242-244
package, adding 241
package, configuring 241
package, executing 248
project, building 248
project creating, with two packages 240

Excel file

BI project, creating 100
Character Map transformation, adding 102-106
data, transferring 99
data flow task, adding 100
DataReader's connection manager, configuring 100
DataReader source, configuring 102
Excel destination, adding 106
Excel destination component, configuring 107-109
package, testing 110
path establishing, from Character Map to Excel destination 106

F

- FileSystemObject** 187
- File System Task, SSIS**
 - BI project, creating 188
 - configuring 189-191
 - Control Flow task, adding 189
 - file, copying from one folder to another 188
 - file, sending by Send Mail task 188
 - operations 188
 - Send Mail task, adding 192-194
 - steps 188
- File Transfer Protocol.** *See* **FTP task**
- FTP task**
 - about 251
 - BI project, creating 253-256
 - building 257
 - default package, renaming 253-256
 - disabling 257-260
 - executing 257
 - files, transferring 251
 - new FTP task, adding 257-260
 - new FTP task, building 261
 - new FTP task, configuring 257-260
 - new FTP task, executing 261
 - package, creating 252, 253

M

- Maintenance Plan task, SSIS**
 - backup database task 289
 - backups 287
 - database, backing up 289
 - database maintenance task 287
 - integrity 287
 - Microsoft SQL server management studio 287, 288
 - optimizations 287
 - package, creating 289-295
- Ms Access Database**
 - about 111
 - BI project, creating 112
 - data, transferring 111
 - data, viewing 119-121
 - data flow, monitoring 119-121
 - data flow task, adding 112
 - DataReaders connection manager, configuring 112

- DataReader source, configuring 112, 113
- data viewer, incorporating 119-121
- OLE DB destination, adding 113, 114
- OLE DB destination component, configuring 114-119
- path establishing, from DataReader component 113, 114

O

- objects, SSIS**
 - connection managers 22, 23
 - connection managers, partial listing 23, 24
 - control flow elements 10
 - data flow components 12
 - Event handlers 24, 25
 - log providers 25
 - log providers, types 26
 - SSIS package 10
 - variables 24
- OLE DB data destination**
 - adding, to canvas 168
 - column, mappings 171
 - configuring 168
 - connection manager, setting up 169, 170
 - data loading errors, handling 171, 172
 - path establishing, from Data Conversion Data Flow transformation 168
 - table, displaying 170
- Oracle 10G XE Server**
 - about 200
 - BI project, creating 203
 - data flow task, adding 203
 - object browser, using 201, 202
 - OLE DB connection manager, configuring 204
 - OLE DB source, adding 204
 - OLE DB source, configuring 205
 - package, building 212
 - package, executing 212
 - path establishing, from OLEDB source to SQL Server destination 209
 - SQL Server destination, adding 207
 - SQL Server destination component, configuring 210-212
 - start mode 200
 - steps 200

stop mode 200
View transferring, to SQL Server 2005 database 200

P

package design, canvas

about 37
control flow 38
data flow 39, 40
event handler 41, 42
items, adding in toolbox 44
package explorer 43, 44
toolbox 44
toolbox items 47

Precedence Constraint

about 195
adding 196
package, building 197
package, executing 197
using, to send mail 195

S

Script task, SSIS

about 273
calculating, variables used 278, 279
data, copying to file 284, 285
data, displaying 284, 285
data, retrieving 282-284
line items 275
package, creating 274
simple calculation, example 274-277
string, building 280, 281

Send Mail Task 75

Simple Mail Transfer Protocol.

See SMTP server

SMO connections 227

SMTP server

about 75
email, sending 75-83
ISPs SMTP server, finding 84

solution explorer

about 48-52

SQL Management Object connections 227

SQL Server Integration Services. *See SSIS*

SSIS

about 7, 8

ActiveX Script task 263
BI project 28
data flow components 239
debugging feature 26
diagnostic feature 26
event handler 239
export wizard, invoking 72, 73
features 26
FileSystemObject 187
File System task 187
import wizard, invoking 72, 73
Maintenance Plan task 287
objects 9
package, importing 69-72
Script task 273
tasks 10, 12
Transfer Database task 228
windows, getting acquainted with 69

SSIS package

creating, to access files 187
creating, to access folders 187
creating, with XML task 175
XMLTask 175

T

Text file

BI project, creating 88
connection manager for DataReader,
 adding 88-90
data, transferring 87
data flow task, adding 88
DataReader source, configuring 90-93
Flat file destination, adding 93
Flat file destination component, configuring
 94-97
package, building 97
package, executing 97
path establishing, from DataReader
 source 93

Transfer Database task

about 227
adding 229
BI project, building 236
BI project, creating 229
configuring 229, 230
connections, establishing 231, 232

database transferring, from one SQL server to another 227
database transferring, from SQL server 2000 to SQL server 2005 227
database transferring, ways 227
destination database, configuring 235
network share, creating 228
package, creating 228
package, executing 236
package, testing 228
SMO connections 227
source database node, configuring 233, 234

W

Web Services Description Language
about 213
miles into kilometers, converting 214

Web Service task, SSIS

about 213
adding 219, 220
BI project, building 225
BI project, creating 219, 220
configuring 220-225
features 213
package, creating 214
package, executing 225
package, testing 214
Visual Studio 2005 solution, creating 214, 215
web service, creating 215, 216
WSDL file, creating 217, 218

windows

about 53, 54

Bookmark window 55
Class View window 56
Code Definition window 56
Debug window 65
Error List window 58
Find Results window 60
Object Browser window 56-58
Other Windows 63, 65
Output window 58
Properties window 58
Property Pages window 67
Server Explorer window 54
solution explorer 48-52
Tasks List window 60
Toolbox window 60

WSDL 213

X

XML documents 176

XMLTask, SSIS
about 175
Diff task 175, 176
merge task 176
task types 175
validate task 176
XPATH task 176
XSLT task 176

XSLT, XMLTask type

documents 178
exercise 184, 185



Thank you for buying **SQL Server Integration Services** **Using Visual Studio 2005**

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

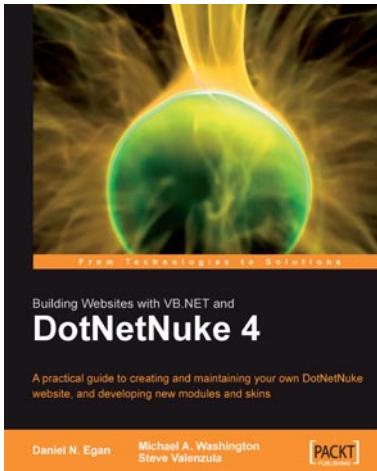
Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to authors@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

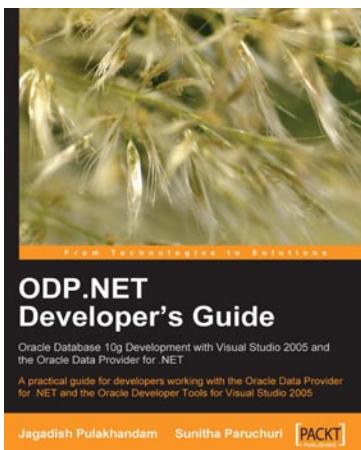


Building Websites with VB.NET and DotNetNuke 4

ISBN: 1-904811-99-X Paperback: 350 pages

A practical guide to creating and maintaining your own DotNetNuke website, and developing new modules and skins

1. Specially revised and updated version of this acclaimed DotNetNuke book
2. Create and manage your own website with DotNetNuke
3. Customize and enhance your site with skins and custom modules
4. Extensive coverage of the DAL and DAL+ for custom module development



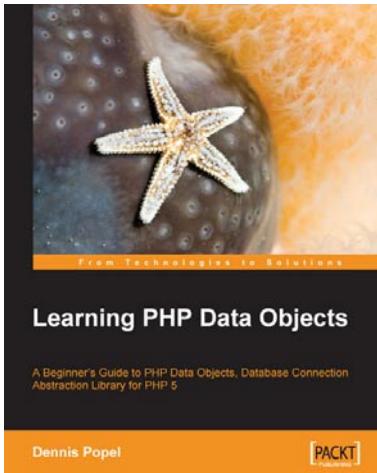
ODP.NET Developer's Guide

ISBN: 978-1-847191-96-0 Paperback: 328 pages

A practical guide for developers working with the Oracle Data Provider for .NET and the Oracle Developer Tools for Visual Studio 2005

1. Application development with ODP.NET
2. Dealing with XML DB using ODP.NET
3. Oracle Developer Tools for Visual Studio .NET

Please check www.PacktPub.com for information on our titles

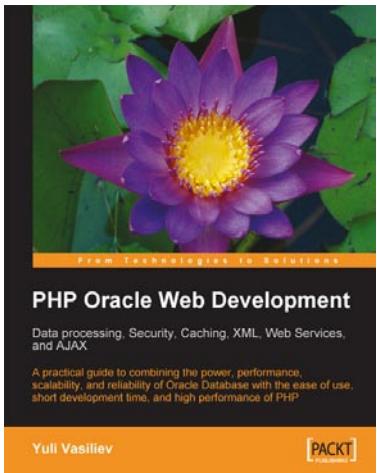


Learning PHP Data Objects

ISBN: 978-1-847192-66-0 Paperback: 200 pages

A Beginner's Guide to PHP Data Objects, Database Connection Abstraction Library for PHP 5

1. An overview of PDO
2. Creating a database and connecting to it
3. Error Handling
4. Error Handling



PHP Oracle Web Development

ISBN: 978-1-847193-63-6 Paperback: 350 pages

A practical guide to combining the power, performance, scalability, and reliability of the Oracle Database with the ease of use, short development time, and high performance of PHP

1. Program your own PHP/Oracle application
2. Move data processing inside the database
3. Distribute data processing between the web/ PHP and Oracle database servers
4. Create reusable building blocks for PHP/ Oracle solutions
5. Use up-to-date technologies, such as Ajax and web services, in PHP Oracle development

Please check www.PacktPub.com for information on our titles