

Microsoft®

OFFICIAL MICROSOFT LEARNING PRODUCT

6235A

**Implementing and Maintaining
Microsoft® SQL Server® 2008
Integration Services**



Be sure to access the extended learning content on your
Course Companion CD enclosed on the back cover of the book.

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2008 Microsoft Corporation. All rights reserved.

Microsoft, Access, Active Directory, ActiveX, Excel, Internet Explorer, Microsoft Press, MSDN, PowerPoint, SQL Server, Visual Basic, Visual C#, Visual Studio, Win32, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are property of their respective owners.

Product Number: 6235A

Part Number: X15-01565

Released: 11/2008

MICROSOFT LICENSE TERMS

OFFICIAL MICROSOFT LEARNING PRODUCTS - TRAINER EDITION – Pre-Release and Final Release Versions

These license terms are an agreement between Microsoft Corporation and you. Please read them. They apply to the Licensed Content named above, which includes the media on which you received it, if any. The terms also apply to any Microsoft

- updates,
- supplements,
- Internet-based services, and
- support services

for this Licensed Content, unless other terms accompany those items. If so, those terms apply.

By using the Licensed Content, you accept these terms. If you do not accept them, do not use the Licensed Content.

If you comply with these license terms, you have the rights below.

1. DEFINITIONS.

- a. **"Academic Materials"** means the printed or electronic documentation such as manuals, workbooks, white papers, press releases, datasheets, and FAQs which may be included in the Licensed Content.
- b. **"Authorized Learning Center(s)"** means a Microsoft Certified Partner for Learning Solutions location, an IT Academy location, or such other entity as Microsoft may designate from time to time.
- c. **"Authorized Training Session(s)"** means those training sessions authorized by Microsoft and conducted at or through Authorized Learning Centers by a Trainer providing training to Students solely on Official Microsoft Learning Products (formerly known as Microsoft Official Curriculum or "MOC") and Microsoft Dynamics Learning Products (formerly known as Microsoft Business Solutions Courseware). Each Authorized Training Session will provide training on the subject matter of one (1) Course.
- d. **"Course"** means one of the courses using Licensed Content offered by an Authorized Learning Center during an Authorized Training Session, each of which provides training on a particular Microsoft technology subject matter.
- e. **"Device(s)"** means a single computer, device, workstation, terminal, or other digital electronic or analog device.
- f. **"Licensed Content"** means the materials accompanying these license terms. The Licensed Content may include, but is not limited to, the following elements: (i) Trainer Content, (ii) Student Content, (iii) classroom setup guide, and (iv) Software. There are different and separate components of the Licensed Content for each Course.
- g. **"Software"** means the Virtual Machines and Virtual Hard Disks, or other software applications that may be included with the Licensed Content.
- h. **"Student(s)"** means a student duly enrolled for an Authorized Training Session at your location.

- i. **"Student Content"** means the learning materials accompanying these license terms that are for use by Students and Trainers during an Authorized Training Session. Student Content may include labs, simulations, and courseware files for a Course.
- j. **"Trainer(s)"** means a) a person who is duly certified by Microsoft as a Microsoft Certified Trainer and b) such other individual as authorized in writing by Microsoft and has been engaged by an Authorized Learning Center to teach or instruct an Authorized Training Session to Students on its behalf.
- k. **"Trainer Content"** means the materials accompanying these license terms that are for use by Trainers and Students, as applicable, solely during an Authorized Training Session. Trainer Content may include Virtual Machines, Virtual Hard Disks, Microsoft PowerPoint files, instructor notes, and demonstration guides and script files for a Course.
- l. **"Virtual Hard Disks"** means Microsoft Software that is comprised of virtualized hard disks (such as a base virtual hard disk or differencing disks) for a Virtual Machine that can be loaded onto a single computer or other device in order to allow end-users to run multiple operating systems concurrently. For the purposes of these license terms, Virtual Hard Disks will be considered "Trainer Content".
- m. **"Virtual Machine"** means a virtualized computing experience, created and accessed using Microsoft® Virtual PC or Microsoft® Virtual Server software that consists of a virtualized hardware environment, one or more Virtual Hard Disks, and a configuration file setting the parameters of the virtualized hardware environment (e.g., RAM). For the purposes of these license terms, Virtual Hard Disks will be considered "Trainer Content".
- n. **"you"** means the Authorized Learning Center or Trainer, as applicable, that has agreed to these license terms.

2. OVERVIEW.

Licensed Content. The Licensed Content includes Software, Academic Materials (online and electronic), Trainer Content, Student Content, classroom setup guide, and associated media.

License Model. The Licensed Content is licensed on a per copy per Authorized Learning Center location or per Trainer basis.

3. INSTALLATION AND USE RIGHTS.

- a. **Authorized Learning Centers and Trainers: For each Authorized Training Session, you may:**
 - i. either install individual copies of the relevant Licensed Content on classroom Devices only for use by Students enrolled in and the Trainer delivering the Authorized Training Session, provided that the number of copies in use does not exceed the number of Students enrolled in and the Trainer delivering the Authorized Training Session, **OR**
 - ii. install one copy of the relevant Licensed Content on a network server only for access by classroom Devices and only for use by Students enrolled in and the Trainer delivering the Authorized Training Session, provided that the number of Devices accessing the Licensed Content on such server does not exceed the number of Students enrolled in and the Trainer delivering the Authorized Training Session.
 - iii. and allow the Students enrolled in and the Trainer delivering the Authorized Training Session to use the Licensed Content that you install in accordance with (ii) or (ii) above during such Authorized Training Session in accordance with these license terms.

- i. Separation of Components. The components of the Licensed Content are licensed as a single unit. You may not separate the components and install them on different Devices.
- ii. Third Party Programs. The Licensed Content may contain third party programs. These license terms will apply to the use of those third party programs, unless other terms accompany those programs.

b. **Trainers:**

- i. Trainers may Use the Licensed Content that you install or that is installed by an Authorized Learning Center on a classroom Device to deliver an Authorized Training Session.
- ii. Trainers may also Use a copy of the Licensed Content as follows:
 - A. Licensed Device. The licensed Device is the Device on which you Use the Licensed Content. You may install and Use one copy of the Licensed Content on the licensed Device solely for your own personal training Use and for preparation of an Authorized Training Session.
 - B. Portable Device. You may install another copy on a portable device solely for your own personal training Use and for preparation of an Authorized Training Session.

4. **PRE-RELEASE VERSIONS.** If this is a pre-release ("beta") version, in addition to the other provisions in this agreement, these terms also apply:

- a. **Pre-Release Licensed Content.** This Licensed Content is a pre-release version. It may not contain the same information and/or work the way a final version of the Licensed Content will. We may change it for the final, commercial version. We also may not release a commercial version. You will clearly and conspicuously inform any Students who participate in each Authorized Training Session of the foregoing; and, that you or Microsoft are under no obligation to provide them with any further content, including but not limited to the final released version of the Licensed Content for the Course.
- b. **Feedback.** If you agree to give feedback about the Licensed Content to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software, Licensed Content, or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.
- c. **Confidential Information.** The Licensed Content, including any viewer, user interface, features and documentation that may be included with the Licensed Content, is confidential and proprietary to Microsoft and its suppliers.
 - i. **Use.** For five years after installation of the Licensed Content or its commercial release, whichever is first, you may not disclose confidential information to third parties. You may disclose confidential information only to your employees and consultants who need to know the information. You must have written agreements with them that protect the confidential information at least as much as this agreement.
 - ii. **Survival.** Your duty to protect confidential information survives this agreement.
 - iii. **Exclusions.** You may disclose confidential information in response to a judicial or governmental order. You must first give written notice to Microsoft to allow it to seek a

protective order or otherwise protect the information. Confidential information does not include information that

- becomes publicly known through no wrongful act;
 - you received from a third party who did not breach confidentiality obligations to Microsoft or its suppliers; or
 - you developed independently.
- d. **Term.** The term of this agreement for pre-release versions is (i) the date which Microsoft informs you is the end date for using the beta version, or (ii) the commercial release of the final release version of the Licensed Content, whichever is first ("beta term").
- e. **Use.** You will cease using all copies of the beta version upon expiration or termination of the beta term, and will destroy all copies of same in the possession or under your control and/or in the possession or under the control of any Trainers who have received copies of the pre-released version.
- f. **Copies.** Microsoft will inform Authorized Learning Centers if they may make copies of the beta version (in either print and/or CD version) and distribute such copies to Students and/or Trainers. If Microsoft allows such distribution, you will follow any additional terms that Microsoft provides to you for such copies and distribution.

5. ADDITIONAL LICENSING REQUIREMENTS AND/OR USE RIGHTS.

a. Authorized Learning Centers and Trainers:

- i. Software.
- ii. **Virtual Hard Disks.** The Licensed Content may contain versions of Microsoft XP, Microsoft Windows Vista, Windows Server 2003, Windows Server 2008, and Windows 2000 Advanced Server and/or other Microsoft products which are provided in Virtual Hard Disks.

A. If the Virtual Hard Disks and the labs are launched through the Microsoft Learning Lab Launcher, then these terms apply:

Time-Sensitive Software. If the Software is not reset, it will stop running based upon the time indicated on the install of the Virtual Machines (between 30 and 500 days after you install it). You will not receive notice before it stops running. You may not be able to access data used or information saved with the Virtual Machines when it stops running and may be forced to reset these Virtual Machines to their original state. You must remove the Software from the Devices at the end of each Authorized Training Session and reinstall and launch it prior to the beginning of the next Authorized Training Session.

B. If the Virtual Hard Disks require a product key to launch, then these terms apply:

Microsoft will deactivate the operating system associated with each Virtual Hard Disk. Before installing any Virtual Hard Disks on classroom Devices for use during an Authorized Training Session, you will obtain from Microsoft a product key for the operating system software for the Virtual Hard Disks and will activate such Software with Microsoft using such product key.

C. These terms apply to all Virtual Machines and Virtual Hard Disks:

You may only use the Virtual Machines and Virtual Hard Disks if you comply with the terms and conditions of this agreement and the following security requirements:

- You may not install Virtual Machines and Virtual Hard Disks on portable Devices or Devices that are accessible to other networks.
- You must remove Virtual Machines and Virtual Hard Disks from all classroom Devices at the end of each Authorized Training Session, except those held at Microsoft Certified Partners for Learning Solutions locations.
- You must remove the differencing drive portions of the Virtual Hard Disks from all classroom Devices at the end of each Authorized Training Session at Microsoft Certified Partners for Learning Solutions locations.
- You will ensure that the Virtual Machines and Virtual Hard Disks are not copied or downloaded from Devices on which you installed them.
- You will strictly comply with all Microsoft instructions relating to installation, use, activation and deactivation, and security of Virtual Machines and Virtual Hard Disks.
- You may not modify the Virtual Machines and Virtual Hard Disks or any contents thereof.
- You may not reproduce or redistribute the Virtual Machines or Virtual Hard Disks.

- ii. **Classroom Setup Guide.** You will assure any Licensed Content installed for use during an Authorized Training Session will be done in accordance with the classroom set-up guide for the Course.
- iii. **Media Elements and Templates.** You may allow Trainers and Students to use images, clip art, animations, sounds, music, shapes, video clips and templates provided with the Licensed Content solely in an Authorized Training Session. If Trainers have their own copy of the Licensed Content, they may use Media Elements for their personal training use.
- iv. **iv Evaluation Software.** Any Software that is included in the Student Content designated as "Evaluation Software" may be used by Students solely for their personal training outside of the Authorized Training Session.

b. **Trainers Only:**

- i. **Use of PowerPoint Slide Deck Templates.** The Trainer Content may include Microsoft PowerPoint slide decks. Trainers may use, copy and modify the PowerPoint slide decks only for providing an Authorized Training Session. If you elect to exercise the foregoing, you will agree or ensure Trainer agrees: (a) that modification of the slide decks will not constitute creation of obscene or scandalous works, as defined by federal law at the time the work is created; and (b) to comply with all other terms and conditions of this agreement.
- ii. **Use of Instructional Components in Trainer Content.** For each Authorized Training Session, Trainers may customize and reproduce, in accordance with the MCT Agreement, those portions of the Licensed Content that are logically associated with instruction of the Authorized Training Session. If you elect to exercise the foregoing rights, you agree or ensure the Trainer agrees: (a) that any of these customizations or reproductions will only be used for providing an Authorized Training Session and (b) to comply with all other terms and conditions of this agreement.

iii. Academic Materials. If the Licensed Content contains Academic Materials, you may copy and use the Academic Materials. You may not make any modifications to the Academic Materials and you may not print any book (either electronic or print version) in its entirety. If you reproduce any Academic Materials, you agree that:

- The use of the Academic Materials will be only for your personal reference or training use
- You will not republish or post the Academic Materials on any network computer or broadcast in any media;
- You will include the Academic Material's original copyright notice, or a copyright notice to Microsoft's benefit in the format provided below:

Form of Notice:

© 2008 Reprinted for personal reference use only with permission by Microsoft Corporation. All rights reserved.

Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the US and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

- 6. INTERNET-BASED SERVICES.** Microsoft may provide Internet-based services with the Licensed Content. It may change or cancel them at any time. You may not use these services in any way that could harm them or impair anyone else's use of them. You may not use the services to try to gain unauthorized access to any service, data, account or network by any means.
- 7. SCOPE OF LICENSE.** The Licensed Content is licensed, not sold. This agreement only gives you some rights to use the Licensed Content. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the Licensed Content only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the Licensed Content that only allow you to use it in certain ways. You may not
- install more copies of the Licensed Content on classroom Devices than the number of Students and the Trainer in the Authorized Training Session;
 - allow more classroom Devices to access the server than the number of Students enrolled in and the Trainer delivering the Authorized Training Session if the Licensed Content is installed on a network server;
 - copy or reproduce the Licensed Content to any server or location for further reproduction or distribution;
 - disclose the results of any benchmark tests of the Licensed Content to any third party without Microsoft's prior written approval;
 - work around any technical limitations in the Licensed Content;
 - reverse engineer, decompile or disassemble the Licensed Content, except and only to the extent that applicable law expressly permits, despite this limitation;
 - make more copies of the Licensed Content than specified in this agreement or allowed by applicable law, despite this limitation;
 - publish the Licensed Content for others to copy;

- transfer the Licensed Content, in whole or in part, to a third party;
- access or use any Licensed Content for which you (i) are not providing a Course and/or (ii) have not been authorized by Microsoft to access and use;
- rent, lease or lend the Licensed Content; or
- use the Licensed Content for commercial hosting services or general business purposes.
- Rights to access the server software that may be included with the Licensed Content, including the Virtual Hard Disks does not give you any right to implement Microsoft patents or other Microsoft intellectual property in software or devices that may access the server.

8. EXPORT RESTRICTIONS. The Licensed Content is subject to United States export laws and regulations. You must comply with all domestic and international export laws and regulations that apply to the Licensed Content. These laws include restrictions on destinations, end users and end use. For additional information, see www.microsoft.com/exporting.

9. NOT FOR RESALE SOFTWARE/LICENSED CONTENT. You may not sell software or Licensed Content marked as "NFR" or "Not for Resale."

10. ACADEMIC EDITION. You must be a "Qualified Educational User" to use Licensed Content marked as "Academic Edition" or "AE." If you do not know whether you are a Qualified Educational User, visit www.microsoft.com/education or contact the Microsoft affiliate serving your country.

11. TERMINATION. Without prejudice to any other rights, Microsoft may terminate this agreement if you fail to comply with the terms and conditions of these license terms. In the event your status as an Authorized Learning Center or Trainer a) expires, b) is voluntarily terminated by you, and/or c) is terminated by Microsoft, this agreement shall automatically terminate. Upon any termination of this agreement, you must destroy all copies of the Licensed Content and all of its component parts.

12. ENTIRE AGREEMENT. This agreement, and the terms for supplements, updates, Internet-based services and support services that you use, are the entire agreement for the Licensed Content and support services.

13. APPLICABLE LAW.

- United States.** If you acquired the Licensed Content in the United States, Washington state law governs the interpretation of this agreement and applies to claims for breach of it, regardless of conflict of laws principles. The laws of the state where you live govern all other claims, including claims under state consumer protection laws, unfair competition laws, and in tort.
- Outside the United States.** If you acquired the Licensed Content in any other country, the laws of that country apply.

14. LEGAL EFFECT. This agreement describes certain legal rights. You may have other rights under the laws of your country. You may also have rights with respect to the party from whom you acquired the Licensed Content. This agreement does not change your rights under the laws of your country if the laws of your country do not permit it to do so.

15. DISCLAIMER OF WARRANTY. The Licensed Content is licensed "as-is." You bear the risk of using it. Microsoft gives no express warranties, guarantees or conditions. You may have additional consumer rights under your local laws which this agreement cannot change. To the extent permitted under your local laws, Microsoft excludes the implied warranties of merchantability, fitness for a particular purpose and non-infringement.

16. LIMITATION ON AND EXCLUSION OF REMEDIES AND DAMAGES. YOU CAN RECOVER FROM MICROSOFT AND ITS SUPPLIERS ONLY DIRECT DAMAGES UP TO U.S. \$5.00. YOU CANNOT RECOVER ANY OTHER DAMAGES, INCLUDING CONSEQUENTIAL, LOST PROFITS, SPECIAL, INDIRECT OR INCIDENTAL DAMAGES.

This limitation applies to

- anything related to the Licensed Content, software, services, content (including code) on third party Internet sites, or third party programs; and
- claims for breach of contract, breach of warranty, guarantee or condition, strict liability, negligence, or other tort to the extent permitted by applicable law.

It also applies even if Microsoft knew or should have known about the possibility of the damages. The above limitation or exclusion may not apply to you because your country may not allow the exclusion or limitation of incidental, consequential or other damages.

Please note: As this Licensed Content is distributed in Quebec, Canada, some of the clauses in this agreement are provided below in French.

Remarque : Ce le contenu sous licence étant distribué au Québec, Canada, certaines des clauses dans ce contrat sont fournies ci-dessous en français.

EXONÉRATION DE GARANTIE. Le contenu sous licence visé par une licence est offert « tel quel ». Toute utilisation de ce contenu sous licence est à votre seule risque et péril. Microsoft n'accorde aucune autre garantie expresse. Vous pouvez bénéficier de droits additionnels en vertu du droit local sur la protection dues consommateurs, que ce contrat ne peut modifier. La ou elles sont permises par le droit locale, les garanties implicites de qualité marchande, d'adéquation à un usage particulier et d'absence de contrefaçon sont exclues.

LIMITATION DES DOMMAGES-INTÉRÊTS ET EXCLUSION DE RESPONSABILITÉ POUR LES DOMMAGES. Vous pouvez obtenir de Microsoft et de ses fournisseurs une indemnisation en cas de dommages directs uniquement à hauteur de 5,00 \$ US. Vous ne pouvez prétendre à aucune indemnisation pour les autres dommages, y compris les dommages spéciaux, indirects ou accessoires et pertes de bénéfices.

Cette limitation concerne:

- tout ce qui est relié au le contenu sous licence , aux services ou au contenu (y compris le code) figurant sur des sites Internet tiers ou dans des programmes tiers ; et
- les réclamations au titre de violation de contrat ou de garantie, ou au titre de responsabilité stricte, de négligence ou d'une autre faute dans la limite autorisée par la loi en vigueur.

Elle s'applique également, même si Microsoft connaissait ou devrait connaître l'éventualité d'un tel dommage. Si votre pays n'autorise pas l'exclusion ou la limitation de responsabilité pour les dommages indirects, accessoires ou de quelque nature que ce soit, il se peut que la limitation ou l'exclusion ci-dessus ne s'appliquera pas à votre égard.

EFFET JURIDIQUE. Le présent contrat décrit certains droits juridiques. Vous pourriez avoir d'autres droits prévus par les lois de votre pays. Le présent contrat ne modifie pas les droits que vous confèrent les lois de votre pays si celles-ci ne le permettent pas.

Acknowledgement

Microsoft Learning would like to acknowledge and thank the following for their contribution towards developing this title. Their effort at various stages in the development has ensured that you have a good classroom experience.

Peter Lammers – Content Developer

Peter Lammers joined Aeshen in 2002 as a Product Analyst, and he has been a Lead Product Analyst since 2005, working on Microsoft TechNet Content, Webcasts, White Papers, and Microsoft Learning Courses. Prior to that he has been a computer programmer and network technician with a 14-year background in troubleshooting, training, modifying and supporting a software application; network administration, troubleshooting, and server, desktop, and firewall support.

Joel Barker – Content Developer

Joel Barker has been developing content for Microsoft server products for five years; prior to that he has held a variety of positions in the IT industry.

Seth Wolf – Content Developer

Seth Wolf has been working with computing technology for over 20 years. His background includes programming, database design, Web site design, network management, hardware troubleshooting, and user support. He remembers the good old days of dBase and Btrieve.

Sunni Brock – Content Developer

Sunni has been working with Aeshen as a content developer since 2006. In her 20-year career, she spent 15 years at Microsoft as a Lead Program Manager in the Windows Product Group and as a Technical Support lead replicating customer configuration scenarios. Prior to joining Aeshen, she served as a Technical Account Manager for Sonic Solutions/Roxio acting as liaison to Adobe, Microsoft, Sony, and other technology leaders.

Karl Middlebrooks - Subject Matter Expert

Mr. Middlebrooks is a Product Analyst with Aeshen, and joined in 2004. He has over 20 years experience in IT and Operations management, network administration, and database administration.

Christopher Randall – Technical Reviewer

MCT, MCDBA, MCITP

With 18 years in IT, Chris Randall has been a trainer, a developer, a database administrator, a courseware author and a consultant. He has worked with and trained customers in the US, Canada, Bermuda and Australia, recently settling down as Senior Trainer for Ameriteach. Chris focuses on SQL Server development and administration, including Microsoft's Business Intelligence toolset. He has spoken at several industry conferences, including SQL PASS and MCTCon.

A graduate of Princeton University, Chris lives in Colorado's Front Range with his family, pets and guitars.

Contents

Module 1: Introduction to SQL Server Integration Services

Lesson 1: Overview of SQL Server Integration Services	1-3
Lesson 2: Using Integration Services Tools	1-9
Lab: Using SQL Server Integration Services	1-17

Module 2: Developing Integration Services Solutions

Lesson 1: Creating an Integration Services Solution	2-3
Lesson 2: Using Variables	2-12
Lesson 3: Building and Running a Solution	2-17
Lab: Implementing an Integration Services Solution	2-22

Module 3: Implementing Control Flow

Lesson 1: Control Flow Tasks	3-4
Lesson 2: Control Flow Precedence Constraints	3-16
Lesson 3: Control Flow Containers	3-21
Lab: Implementing Control Flow	3-27

Module 4: Implementing Data Flow

Lesson 1: Data Flow Sources and Destinations	4-3
Lesson 2: Basic Data Flow Transformations	4-11
Lesson 3: Advanced Data Flow Transformations	4-21
Lesson 4: Data Flow Paths	4-32
Lab: Implementing Data Flow	4-40

Module 5: Implementing Logging

Lesson 1: Overview of Integration Services Logging	5-3
Lesson 2: Using Logging	5-13
Lab: Implementing Logging	5-18

Module 6: Debugging and Error Handling

Lesson 1: Debugging a Package	6-3
Lesson 2: Implementing Error Handling	6-13
Lab: Debugging and Error Handling	6-21

Module 7: Implementing Checkpoints and Transactions

Lesson 1: Implementing Checkpoints	7-3
Lesson 2: Implementing Transactions	7-10
Lab: Implementing Checkpoints and Transactions	7-17

Module 8: Configuring and Deploying Packages

Lesson 1: Package Configurations	8-3
Lesson 2: Preparing and Deploying Packages	8-15
Lab: Deploying Packages	8-30

Module 9: Managing and Securing Packages

Lesson 1: Managing Packages	9-3
Lesson 2: Securing Packages	9-17
Lab: Managing and Securing Packages	9-26

Lab Answer Keys

About This Course

This section provides you with a brief description of the course, audience, suggested prerequisites, and course objectives.

Course Description

This three-day instructor-led course teaches students how to implement an Integration Services solution in an organization. The course discusses how to develop, deploy, and manage Integration Services packages.

Audience

This course is intended for information technology (IT) professionals and developers who need to implement data transfer or extract, transform, and load (ETL) solutions by using Microsoft SQL Server 2008 Integration Services.

The secondary audiences for this course are individuals who develop applications that deliver content from SQL Server databases.

Student Prerequisites

This course requires that you meet the following prerequisites:

- Exposure to enterprise data import and export scenarios.
- Experience navigating the Microsoft Windows Server environment.
- Experience with Microsoft SQL Server, including:
 - SQL Server Agent
 - SQL Server query language (SELECT, UPDATE, INSERT, and DELETE)
 - SQL Server system tables
 - SQL Server accounts (users and permissions)

In addition, it is recommended, but not required, that students have completed:

- Course 6231: Maintaining a Microsoft SQL Server 2008 Database
- Course 6232: Implementing a Microsoft SQL Server 2008 Database

Course Objectives

After completing this course, students will be able to:

- Describe SQL Server Integration Services and its tools.
- Create an Integration Services package.
- Implement control flow in an Integration Services package.
- Implement data flow in an Integration Services package.
- Implement logging in an Integration Services package.
- Debug and implement error handling in an Integration Services package.
- Implement checkpoints and transactions in an Integration Services package.
- Deploy an Integration Services package.
- Manage and secure an Integration Services package.

Course Outline

This section provides an outline of the course:

Module 1: Introduction to SQL Server 2008 Integration Services

This module covers the role that Integration Services plays in extracting, transforming, and loading data. It also covers the tools that are used to build and manage Integration Services solutions.

Module 2: Developing Integration Services Solutions

This module covers the development tasks that are involved in creating an Integration Services package.

Module 3: Implementing Control Flow

This module covers the tasks and precedence constraints that can be used to implement control flow in an Integration Services package.

Module 4: Implementing Data Flow

This module covers the data flow sources, transformations, and destinations that can be used to implement a data flow task in an Integration Services control flow. It also explains how to use data flow paths to direct valid and invalid rows through the data flow.

Module 5: Implementing Logging

This module covers how to use logging in an Integration Services package, and explained how to configure and use logging providers to generate information about a package's execution.

Module 6: Debugging and Error Handling

This module covers how to debug Integration Services packages by using the debugging tools in Business Intelligence Development Studio. It then explains how to implement error-handling logic in an Integration Services package.

Module 7: Implementing Checkpoints and Transactions

This module covers what checkpoints are and how to implement them. The module then discusses transactions, and describes how to implement transactional data access logic in an Integration Services package.

Module 8: Configuring and Deploying Packages

This module covers how to create Package Configurations and how to deploy Integration Services packages to production servers.

Module 9: Managing and Securing Packages

This module covers management tasks that relate to Integration Services packages and explained how to perform those tasks by using the Integration Services management tools. It also describes how to secure Integration Services packages.

Course Materials

The following materials are included with your kit:

- *Course Handbook.* A succinct classroom learning guide that provides all the critical technical information in a crisp, tightly-focused format, which is just right for an effective in-class learning experience.
 - Lessons: Guide you through the learning objectives and provide the key points that are critical to the success of the in-class learning experience.
 - Labs: Provide a real-world, hands-on platform for you to apply the knowledge and skills learned in the module.
 - Module Reviews and Takeaways: Provide improved on-the-job reference material to boost knowledge and skills retention.
 - Lab Answer Keys: Provide step-by-step lab solution guidance at your finger tips when it's needed.
- *Course Companion CD.* Searchable, easy-to-navigate digital content with integrated premium on-line resources designed to supplement the Course Handbook.
 - Lessons: Include detailed information for each topic, expanding on the content in the Course Handbook.
 - Labs: Include complete lab exercise information and answer keys in digital form to use during lab time.
 - Resources: Include well-categorized additional resources that give you immediate access to the most up-to-date premium content on TechNet, MSDN®, Microsoft Press®.
 - Student Course Files: Include the Allfiles.exe, a self-extracting executable file that contains all the files required for the labs and demonstrations.

Note: To access the full course content, insert the Course Companion CD into the CD-ROM drive, and then in the root directory of the CD, double-click StartCD.exe.

- *Course evaluation.* At the end of the course, you will have the opportunity to complete an online evaluation to provide feedback on the course, training facility, and instructor.

To provide additional comments or feedback on the course, send e-mail to support@mscourseware.com. To inquire about the Microsoft Certification Program, send e-mail to mcphelp@microsoft.com.

Virtual Machine Environment

This section provides the information for setting up the classroom environment to support the business scenario of the course.

Virtual Machine Configuration

In this course, you will use Microsoft Virtual Server 2005 R2 with SP1 to perform the labs.

Important: At the end of each lab, you must close the virtual machine and must not save any changes. To close a virtual machine without saving the changes, perform the following steps:

1. On the virtual machine, on the **Action** menu, click **Close**.
2. In the **Close** dialog box, in the **What do you want the virtual machine to do?** list, click **Turn off and delete changes**, and then click **OK**.

The following table shows the role of each virtual machine used in this course:

Virtual machine	Role
6235A-NY-SQL-01	Windows Server 2008 with SQL Server 2008

Software Configuration

The following software is installed on each VM:

- Windows Server 2008 Enterprise Edition
- SQL Server 2008

Course Files

There are files associated with the labs in this course. The lab files are located in the folder E:\Labfiles on the student computers.

Classroom Setup

Each classroom computer will have the same virtual machine configured in the same way.

Course Hardware Level

To ensure a satisfactory student experience, Microsoft Learning requires a minimum equipment configuration for trainer and student computers in all Microsoft Certified Partner for Learning Solutions (CPLS) classrooms in which Official Microsoft Learning Product courseware are taught.

This course requires that you have a computer that meets or exceeds hardware level 5.5, which specifies a 2.4-gigahertz (GHz) (minimum) Pentium 4 or equivalent CPU, at least 2 gigabytes (GB) of RAM, 16 megabytes (MB) of video RAM, and two 7200 RPM 40-GB hard disks.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 1

Introduction to SQL Server Integration Services

Contents:

Lesson 1: Overview of SQL Server Integration Services	1-3
Lesson 2: Using Integration Services Tools	1-9
Lab: Using SQL Server Integration Services	1-17

Module Overview

- Overview of SQL Server Integration Services
- Using Integration Services Tools

In this module, you will be introduced to SQL Server Integration Services (SSIS) as it is used with Microsoft® SQL Server® 2008. Integration Services is a tool designed to simplify extract, transform, and load projects. You will use the Import and Export Wizard to quickly create simple packages. Then, you will use Business Intelligence Studio to modify a package. In addition, this module will introduce package management.

You will learn about the architecture of SSIS and how to implement it. When called upon to create complex transformations or to interact with data that is in a different format, knowledge of SSIS will help you to fulfill requirements.

Lesson 1

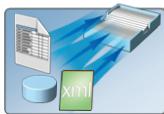
Overview of SQL Server Integration Services

- What Is SQL Server Integration Services?
- Common Uses for Integration Services
- Fundamental Integration Services Concepts
- Integration Services Architecture

In this lesson, you will be introduced to SQL Server Integration Services (SSIS), including the tools that administrators and developers use for ETL projects. You will learn about how SSIS can be accessed through the SQL Server Management Studio, the Business Intelligence Studio, and through wizards. You will learn about the overall architecture of SSIS as well.

With this knowledge you will be able to identify the utility of SSIS for ETL scenarios that you are likely to encounter.

What Is SQL Server Integration Services?



ETL (extract, transform, and load): The process of collecting and cleaning data from various sources



SSIS is a platform for ETL operations



SSIS consists of a control flow engine and a data flow engine

Key Points

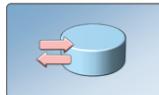
Extract, transform, and load, or ETL, is the process of collecting and cleaning data from various sources.

- SQL Server Integration Services is the SQL Server 2008 tool used to simplify ETL operations. The design goal for SSIS is to reduce manual processes during an ETL project.
- The SSIS architecture has two engines: The control flow engine and the data flow engine.

- The control flow in an Integration Services package is constructed by using different types of control flow elements, such as tasks that support data flow, prepare data, perform workflow and business intelligence functions, and implement scripts.
- The data flow in an Integration Services package is constructed by using different types of data flow elements: sources that extract data, transformations that modify and aggregate data, destinations that load data, and paths that connect the outputs and inputs of data flow components into a data flow.

Question: Have you worked on past ETL projects that involved manual processes?

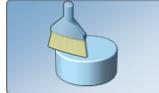
Common Uses for Integration Services



Import and export data



Integrate heterogeneous data



Clean and standardize data



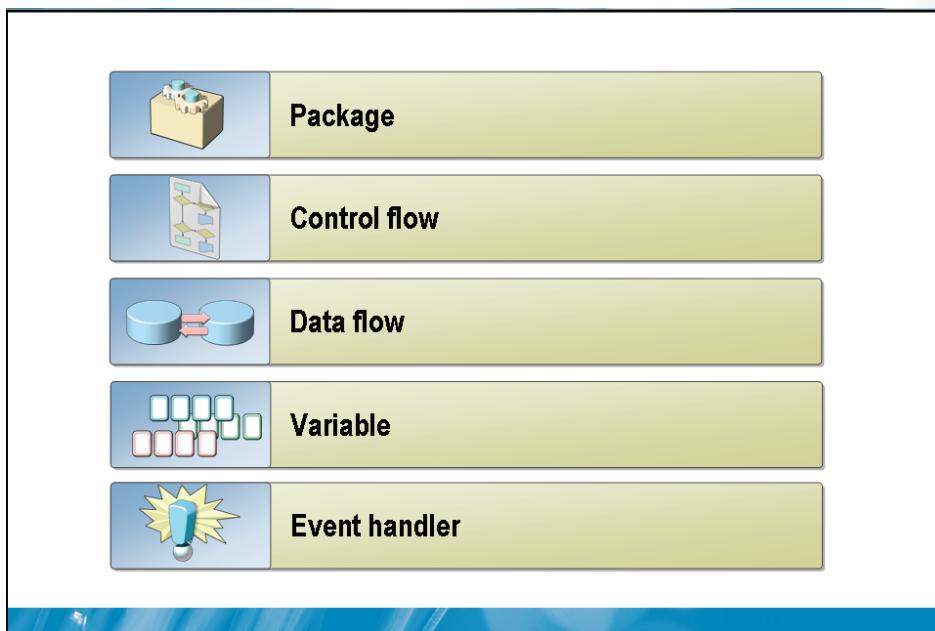
Support BI solutions

Key Points

- SSIS can be used to move data between SQL Server and a variety of sources, including flat files, Oracle databases, and others.
- SSIS can create integration projects to mesh data from multiple sources.
- During ETL projects, data often needs to be cleaned or standardized. The SSIS data flow can include automated cleaning procedures.
- Using lookups and fuzzy lookups, an SSIS package can standardize data such as matching names in two different columns to a client number.
- SSIS is integrated with other business intelligence tools in SQL Server 2008. It will often be an element of larger Business Intelligence (BI) operations.

Question: Have you had issues in the past cleaning and standardizing data during ETL projects?

Fundamental Integration Services Concepts

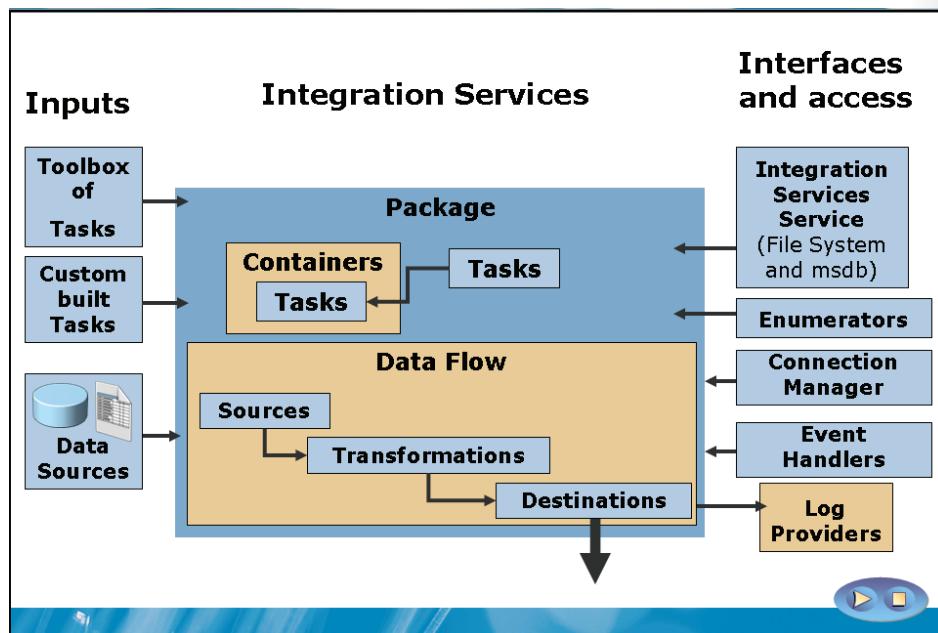


Key Points

The following are the elements that make up an Integration Services project:

- A **package** is the unit of work that is retrieved, executed, and saved. Packages are stored in either the msdb database or in an XML file with the suffix “dtsx”.
- The **Control Flow** contains tasks and containers. Tasks perform operations such as preparing and copying data. Containers are used to manage groupings of tasks as a logical unit.
- The **Data Flow** contains sources, transformations, and destinations.
- **Variables** are used to perform more dynamic operations, such as updating column values and controlling repeated control flows.
- **Event Handlers** run in response to run time events to aid in monitoring and responding to packages.

Integration Services Architecture



Key Points

- The Integration Services service, available in SQL Server Management Studio, monitors running Integration Services packages and manages the storage of packages.
- The Integration Services runtime saves the layout of packages, runs packages, and provides support for logging, breakpoints, configuration, connections, and transactions. The Integration Services run-time executables are the package, containers, and tasks.
- The Data Flow task encapsulates the data flow engine. The data flow engine provides the in-memory buffers that move data from source to destination, and calls the sources that extract data from files and relational databases and manages the transformations that modify data, and the destinations that load data or make data available to other processes.

Lesson 2

Using Integration Services Tools

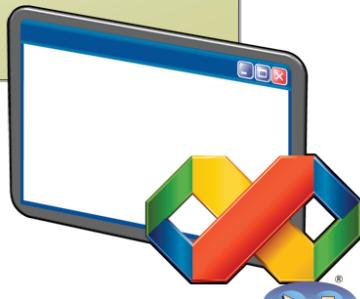
- Introducing Business Intelligence Development Studio
- SSIS in Business Intelligence Studio
- Using Packages
- Using Wizards

In this lesson, you will be introduced to the tools used to create and manage SSIS packages. The interface most frequently used in creating and managing SSIS packages is the Business Intelligence Studio. Built upon the Business Intelligence Studio are several wizards to streamline certain tasks. In addition, SSIS packages can be accessed from command line utilities.

With these tools, you will be able to create SSIS solutions for your ETL projects.

Introducing Business Intelligence Studio

- Based on Microsoft Visual Studio 2008
- Project types specific to business intelligence
- Used to develop for Analysis Services, Reporting Services, and Integration Services
- Includes tools, wizards, and templates



Key Points

The primary tool for creating SSIS solutions is the Business Intelligence Studio.

- Business Intelligence Development Studio is Microsoft® Visual Studio® 2008 with additional project types specific to SQL Server business intelligence.
- Integration Services development tools are built on the Visual Studio platform.

SSIS in Business Intelligence Studio

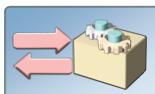
- Five tabs in SSIS Designer**
- Control Flow
 - Data Flow
 - Event Handlers
 - Package Explorer
 - Progress/Execution Results



Key Points

- SSIS Packages are created in or imported into Business Intelligence Studio.
- Use SSIS Designer to develop the control flow, the data flow, connection managers, and event handlers.
- The Control Flow and the Data Flow tabs access the work surface for their respective flows.
- The Event Handler tab is used to create event handlers.
- The Package Explorer tab provides an explorer view of the package, with the package as a container at the top of the hierarchy, and underneath it, the connections, executables, event handlers, log providers, precedence constraints, and variables that you have configured in the package.
- The Progress tab displays information about package execution when you run a package in Business Intelligence Development Studio.

Using Packages



Import and export Integration Services packages



Manage Integration Services packages



Run Integration Services packages



Monitor running Integration Services packages

Key Points

Packages are the executable SSIS process.

- Packages can be run through SQL Server Management Studio which invokes DTExec. Packages can also be run directly from the command line using **DTExec**.
- To view information about current packages on the Summary page of SQL Server Management Studio, click the **Running Packages** folder. Information such as the execution duration of running packages is listed on the Summary page, allowing you to monitor packages at runtime from the interface.
- Through the interface, one can import and export packages from the msdb database.

Question: In addition to the logging and monitoring features built in to SSIS, what other tools might be useable to keep track of package execution?

SSIS Wizards



Key Points

Wizards can be run as a stand alone application or invoked from the Business Intelligence Studio.

- The **Import and Export Wizard** can be used to quickly create a package that will copy data to or from any source that has a .NET or OLE DB database provider, including other SQL databases, flat files and Microsoft® Excel® spreadsheets.
- The **Package Configuration Wizard** guides you through the steps to create the configurations that update the values of properties of packages and package objects at run time.
- Packages can be created on one server and migrated to others. Use the **Package Installation Wizard** to deploy and update packages.

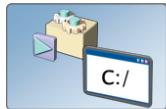
Demonstration: Running an Integration Services Package

In this demonstration, you will learn how to:

- Create a package using the Import and Export Wizard
- Run the package using the Execute Package utility

Question: When would you want to run the package immediately from the wizard rather than saving it and running it later with the execute package utility?

Command Prompt Utilities



DTExec utility



Execute Package Utility (dtexecui)



DTUTIL utility

Key Points

- The **DTExec** command can run packages from a command line or script. All other methods of running a package, including launching them from the SQL Server Management Studio and the Execute Package Utility, form DTExec commands and execute them.
- The **Execute Package Utility** (dtexecui.exe) always runs a package on the local computer, even if the package is saved on a remote server.
- Use **DTUTIL** to manage packages. **DTUTIL** can import and export packages from the msdb database. **DTUTIL** can also sign, delete, and verify packages.

Demonstration: Executing a Package using Line Commands

In this demonstration, you will learn how to:

- Export a package
- Edit a Package using Microsoft Visual Studio
- Use DTUtil to import a package
- Run a package using DTExec

Question: When would direct execution of **DTExec** and **DTUTIL** be preferable to the Management Studio interface for managing and running packages?

Lab: Using SQL Server Integration Services

- Exercise 1: Using the Import and Export Wizard
- Exercise 2: Running an Integration Services Package
- Exercise 3: Import Data Using the Import and Export Wizard
- Exercise 4: Managing Packages

Logon information

Virtual machine	NY-SQL-01
User name	NY-SQL-01\Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Using the Import and Export Wizard

Scenario

The Marketing department at Adventure Works requires a current list of the currencies used in the AdventureWorks2008 database. They want the list in the format of a comma-delimited text file. You must use SQL Server Integration Services to provide this information in the correct format. Create the flat file from the information in the currency table.

In this exercise, you will use the Import and Export Wizard to export the currency table to a flat file.

The main tasks for this exercise are as follows:

1. Connect to the database.
2. Open the Import and Export Wizard.
3. Choose a data source.
4. Choose a destination.

5. Write a T-SQL statement.
6. Configure a flat file destination.
7. Create a package.
8. Save the package.

- ▶ **Task 1: Open SQL Server Management Studio and connect to the database engine on NY-SQL-01**
 - Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
 - Start **Server Management Studio** and connect to the NY-SQL-01 SQL server database engine.
- ▶ **Task 2: Use the SQL Server Import and Export Wizard to export the currency data**
 - For the AdventureWorks2008 database, start the **Export Wizard**.
- ▶ **Task 3: Choose a data source**
 - For data source, specify the following options:
 - Data Source: **SQL Native Client 10.0**
 - Server Name: **NY-SQL-01**
- ▶ **Task 4: Choose a destination**
 - Specify a flat file destination of **E:\Mod01\Labfiles\Currency.txt**, stating that the column names are in the first row.
- ▶ **Task 5: Write a Transact-SQL statement to extract currency data**
 - Use the following query to extract the data from the database:

```
SELECT CurrencyCode, Name FROM Sales.Currency
```

- ▶ **Task 6: Configure the flat file destination**
 - For Row delimiter, use {CR}{LF}.
 - For Column delimiter, use commas.
- ▶ **Task 7: Create an Integration Services package**
 - Do not execute the package immediately. Instead, save it to the database using storage server roles for access control.
- ▶ **Task 8: Save the Integration Services package**
 - Save the package to the msdb database with the name **CurrencyExport**.
 - Write a description for the package that describes what task it performs.
 - Use Windows Authentication.

Results: After this exercise, you should have created a package capable of exporting the Currency table from the database. The package should now be stored in the msdb database as **CurrencyExport**.

Exercise 2: Running an Integration Services Package

Scenario

Now that the package is created, you will execute it and complete the export.

In this exercise you will run the **CurrencyExport** package and confirm that the export completed successfully.

The main tasks for this exercise are as follows:

1. Connect to Integration Services.
2. Launch the Execute Package Utility.
3. Run the currency package.
4. View the exported data.

► **Task 1: Connect to Integration Services**

- In SQL Server Management Studio, connect to Integration Services on NY-SQL-01.

► **Task 2: Launch the Execute Package Utility**

- Locate the **CurrencyExport** package in the packages hierarchy.
- Launch the **Execute Package Utility** for the **CurrencyExport** package.

► **Task 3: Run the Currency Package**

- Run the package.

► **Task 4: View the Currency.txt File**

- Browse to E:\Mod01\Labfiles.
- View the file and confirm the currency.txt file contents.

Results: After this exercise, you should have successfully exported the contents of the currency table to the file **E:\Mod01\Labfiles\Currency.txt**.

Exercise 3: Import Data Using the Import and Export Wizard

Scenario

Use the Import and Export Wizard to import data from a flat file to the SQL database.

In this exercise you will use the **Import and Export Wizard** to move data stored in a flat file to the SQL database.

The main task for this exercise is as follows:

1. Import data to the AdventureWorks database from a flat file.

► Task 1: Import Data Using the Import and Export Wizard

- For the **AdventureWorks** database, start the **Import Wizard**.
- Select a flat file source of **E:\Mod01\Labfiles\Currency.txt**.
- Run the package upon the completion of the wizard.

Results: After this exercise, you should have successfully imported the contents of **E:\Mod01\Labfiles\Currency.txt** to the AdventureWorks database.

Exercise 4: Managing Packages

Scenario

You need to modify the CurrencyExport package for use with a table containing business contacts. You have been asked to export the package to a file system, change elements of the tasks, and then execute the task to export the contacts table to a flat file.

In this exercise you will use the Import and Export Wizard to move data stored in a flat file to the SQL database.

The main tasks for this exercise are as follows:

1. Export a package to the file system using SQL Server Management Studio.
2. Edit a package.
3. Save a package.
4. Use DTUTIL to import a package.
5. Use DTExec to save a package.

► Task 1: Export a package to the file system using SQL Server Management Studio

- In Object Explorer, under NY-SQL-01 (Integration Services 10.0.1300 – NY-SQL-01), right-click **CurrencyExport**, point to **Tasks**, and then click **Export Package**.
- Export the package to **E:\Mod01\Labfiles\CurrencyExport1**.
- Encrypt sensitive data with user key.

► Task 2: Edit a package

- Right-click **CurrencyExport1.dtsx** and then click **Edit**.
- In the **Data Flow** tab, edit the **Source - Query** task.
- Use the following query to select items to export:

```
SELECT Firstname, Lastname
FROM Person.Person
WHERE BusinessEntityID < 20
```

- Parse the query.
- Confirm that Columns **Firstname** and **Lastname** are checked.
- Click **OK**.
- Right-click **Destination – Currency** and then click **Edit**.
- In the **Restore Invalid Column References Editor** dialog box, in the **Available Columns** column, for the first row, select **Source - Query.FirstName**.
- In the **Available Columns** column, for the second row, select **Source - Query.LastName**.
- Right-click **DestinationConnectionFlatFile** and then click **Edit**.
- In the **Flat File Connection Manager Editor** dialog box, in the **File name** field, type **E:\MOD01\Labfiles\Contacts.txt**.
- Click **Advanced**.
- Click **CurrencyCode**.
- In the right column, select **CurrencyCode** and type **FirstName**.
- In the center column, click **Name**.
- In the right column, select **Name** and type **LastName**.
- Click **OK**.
- Edit **Destination - Currency.txt**.
- In the dialog box, click **Yes**.
- In the **Flat File Destination Editor** dialog box, click **OK**.

► **Task 3: Save the package**

- Save a copy of the package to the file system as E:\MOD01\Labfiles>ContactExport.dtsx.

► **Task 4: Import a package using DTUTIL**

- Import the new package to the SQL database using the following line command:
 - dtutil /FILE E:\MOD01\LabFiles\ContactExport.dtsx /COPY SQL;ContactExport
- Confirm that the package is visible in SQL Server Management Studio.

► **Task 5: Run a package using DTExec**

- Use DTExec to execute the ContactExport package.
- Confirm that **Contacts.txt** has been created at E:\MOD01\LabFiles.
- Turn off virtual machine and delete changes.

Results: After this exercise, you should have modified an existing package, imported it into the SQL msdb database, and executed it.

Module Review and Takeaways

- Review Questions
- Best Practices
- Tools

Review Questions

1. When creating a package, what element would you add to trigger an alert when an error occurs?
2. In what scenario would the DTExec and DTUTIL commands be useful?
3. Why would you export a package from the SQL database to the file system?

Best Practices Related to SSIS

Supplement or modify the following best practices for your own work situations:

- Use the Import and Export Wizard for less complex ETL operations.
- Build SSIS packages that minimize manual steps in ETL projects.

Tools

Tool	Use for	Where to find it
Microsoft SQL Server Management Studio	<ul style="list-style-type: none">Managing SQL server databases and tables	Start All Programs Microsoft SQL Server 2008
SQL Server Business Intelligence Development Studio	<ul style="list-style-type: none">Managing SQL server applications	Start All Programs Microsoft SQL Server 2008
DTExec	<ul style="list-style-type: none">Running packages in scripts or through the command line	Type DTExec /? from the command line
DTUTIL	<ul style="list-style-type: none">Moving, checking, and verifying packages	Type DTUTIL /? from the command line

Module 2

Developing Integration Services Solutions

Contents:

Lesson 1: Creating an Integration Services Solution	2-3
Lesson 2: Using Variables	2-12
Lesson 3: Building and Running a Solution	2-17
Lab: Implementing an Integration Services Solution	2-22

Module Overview

- Creating an Integration Services Solution
- Using Variables
- Building and Running a Solution

In this module, you will learn how to create a SQL Server Integration Services (SSIS) solution. You will be introduced to the key elements of SSIS packages and how to utilize them in an Extract, Transform, and Load (ETL) scenario.

By understanding Integration Services packages and their component elements such as variables, you will be able to design solutions for ETL challenges that you face.

SQL Server administrators are often called upon to move and change large amounts of data. One example would be a migration project. It is inevitable that you will need to move the data in one server to another, perhaps different types of servers such as Oracle. Another example would be the creation of regular reports.

In both situations you want a process that requires as few manual steps as possible. In addition, it would be ideal to have a reproducible package. SSIS is designed to make automated, moveable packages that can be executed in a number of ways.

Lesson 1

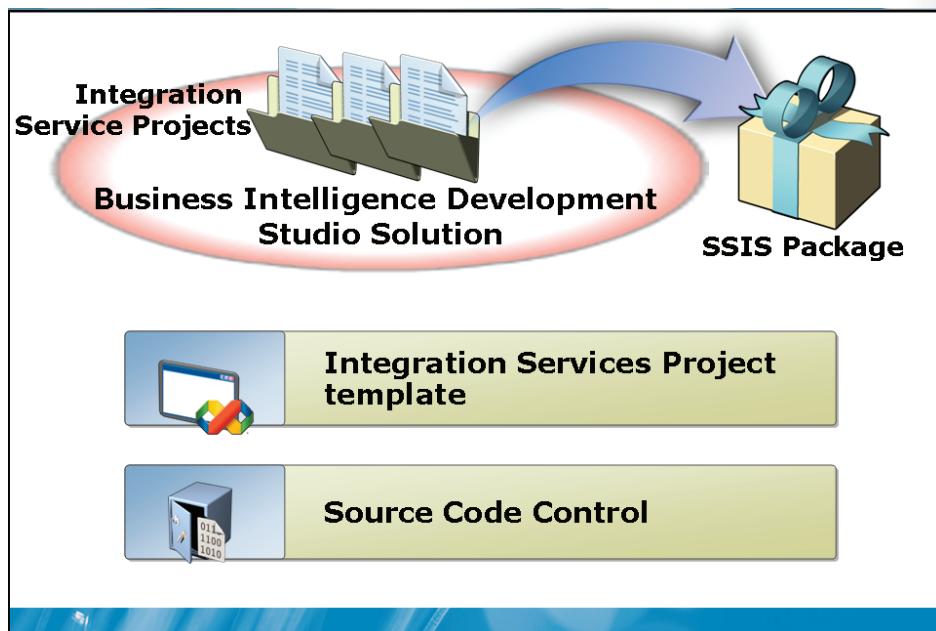
Creating an Integration Services Solution

- Integration Services Projects
- Working with Projects
- Working with Connections
- Implementing Control Flow
- Implementing Data Flow
- Implementing Event Handlers
- Using the Package Explorer

In this lesson, you will look at how the pieces fit together: An SSIS package is the unit that contains a solution. Within a package, the control flow, data flow and event handler design surfaces allow the developer to visually assemble a solution from the elements.

You will see how these elements fit together and how to create the basic parts of a solution.

Integration Services Projects



Key Points

- An Integration Services project is based on the Integration Services template available in Business Intelligence Studio.
- Integration Services projects are contained within a Business Intelligence Development Studio solution.
- The executable unit created by a project is a package.
- A project can be created in a new solution or added to an existing solution.
- When developing, use a version control tool such as Microsoft® Team Foundation Version Control to track changes.

Question: Do you currently use a source or version control tool?

Working with Projects



A new SSIS project contains a default package



You can add multiple packages to the same project



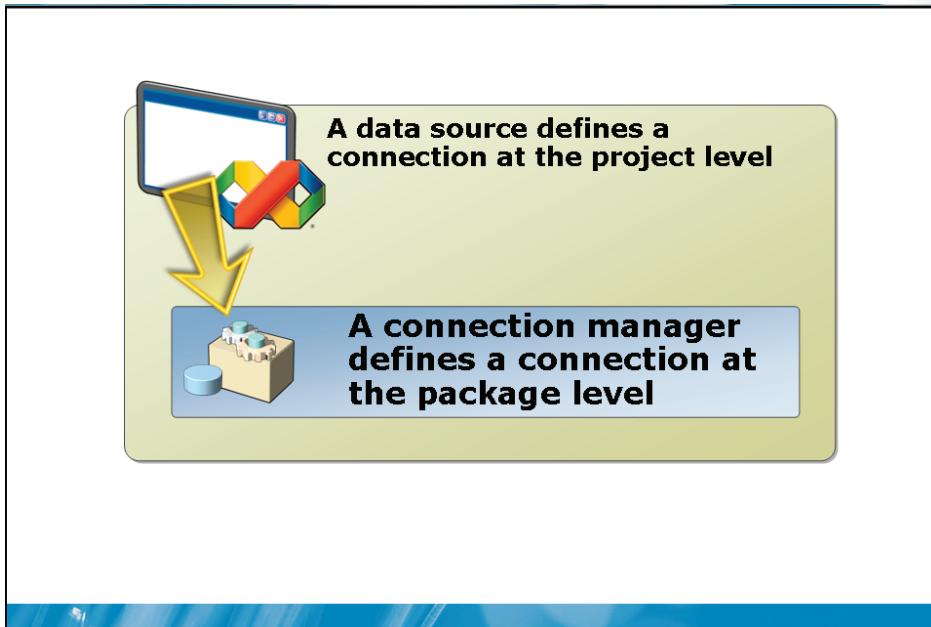
Packages saved to SQL Server, package store, or file system

Key Points

When you create a new SSIS project, the necessary files are created in the default package.

- A package is a collection of connections to data sources, control flow elements, data flow elements, event handlers, variables, and configurations.
- Multiple packages can be contained within a single project. A project is the unit of work during development, not a runtime concept.
- Packages can be saved to three different locations: The msdb system database, the SSIS package store, or to the file system as a dtsx file.

Working with Connections



Key Points

In order to interact with data, SSIS uses connection managers.

- Define data sources at the project level.
- Connection managers are created at the package level and are only available within that package.

Implementing Control Flow

1 Drag components to control flow design surface

2 Use precedence constraints to connect elements

3 Configure properties for each element

4 Group related tasks in containers

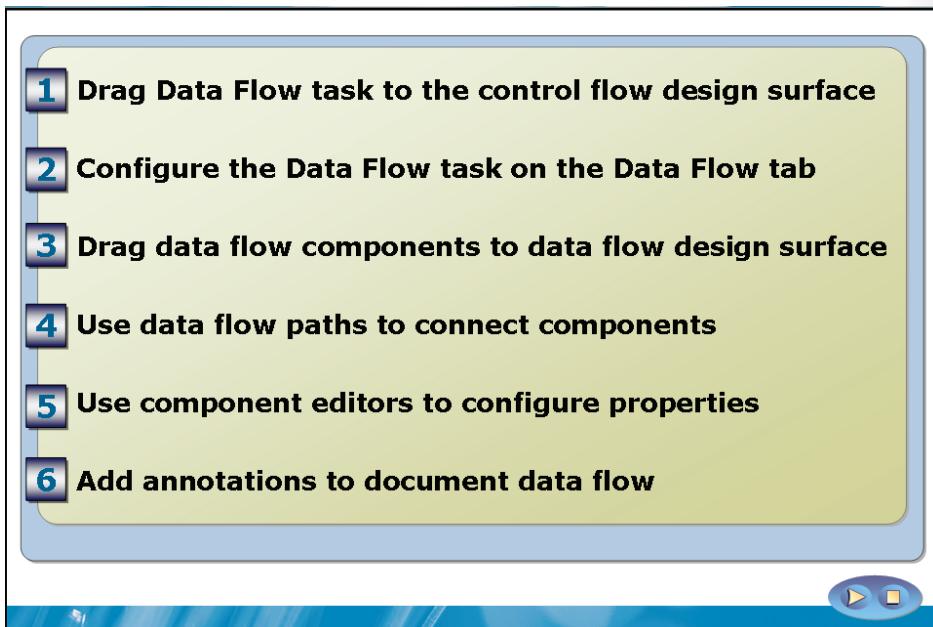
5 Add annotations to document control flow

Key Points

- Control Flow is the definition of a package's workflow.
- Use the SSIS Designer control flow design surface to create control flows.
- Precedence constraints connect elements.
- Tasks can be placed into containers to create logical units.
- Make use of naming and annotation features to clarify a control flow.

Question: Why would you need to put tasks into containers?

Implementing Data Flow



Key Points

Data flows are created on the data flow task design surface.

- A data flow is an element of a control flow.
- Data flow elements are sources, transformations, and destinations.
- Use annotations and naming to clarify the data flow.

Question: Why is the data flow separated from the control flow in SSIS Designer?

Implementing Event Handlers

The diagram illustrates the process of implementing event handlers in a package. It features a yellow callout box containing four steps, a blue callout box listing event types, and a decorative footer bar.

Steps to Implement Event Handlers:

- ✓ Define event handlers on Event Handlers tab
- ✓ Click design surface link to create event handler
- ✓ Drag tasks and containers to design surface
- ✓ Create event handlers at highest logical point

Some Types of Events:

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

▶ ◀

Key Points

Event handlers are used to respond to events during a project.

- Event handlers trigger an action based on a runtime event in the control flow.
- Can be defined for any package, container, or task.

Question: Besides sending a message when an error occurs, when would you implement an event handler?

Using the Package Explorer



Key Points

The Package Explorer tab provides an explorer view of the package.

- **Package Explorer** displays a hierarchical view of the package objects.
- Available objects are tasks, containers, precedence constraints, event handlers, connection managers, log providers, and variables.

Demonstration: Creating a Package

In this demonstration, you will learn how to:

- Create a new Integration Services project
- Add a data source and data source view
- Create a connection manager
- Save the project to the file system

Question: Why would we save the project to the file system instead of to the system database?

Lesson 2

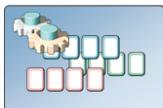
Using Variables

- Introduction to Variables
- Defining Variables
- Using Variables

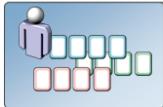
In this lesson, you will learn about variables in Integration Services projects. In many cases, ETL solutions will need to make use of variables. For instance, a message sent by the package from an event handler should include the error number and additional information.

You will see how variables are introduced to an SSIS solution, how to use the Variable pane of Business Intelligence studio, and demonstrate some scenarios in which to implement variables.

Introduction to Variables



SSIS defines a set of system variables for each package



You can define user variables to use throughout your package

Key Points

It is important to understand variables in order to create SSIS solutions.

- Variables are a way to store values that can then be passed among elements in an Integration Services package.
- SSIS includes a set of system variables, such as MachineName and PackageID.
- User variables can be created for packages, Loop containers, Foreach Loop containers, sequence containers, tasks, and event handlers.

Question: How could system variables be utilized with event handlers to send informational messages to administrators?

Defining Variables

- Create variables in the Variables pane**
- Create variables at the correct scope**
- Specify name, data type, and initial value**

Key Points

- In SSIS Designer, variables are created and handled in the Variables pane.
- After creation, a variable scope cannot be changed.
- Specify name, data type, and initial value when creating variables.

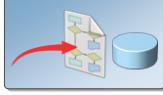
Using Variables



Pass variables in and out of script



Use variables in loops



Pass variables in and out control flow and data flow elements



RowCount transformation

Key Points

Variables are commonly used in the following situations:

- Use variables in a Script task or Script component to pass values in and out of the script.
- Use variables in For Loop and Fore\Each Loop containers to define repeating control flow actions.
- Use variables to pass values into or out of control flow and data flow elements.
- Use variables to track transformations using a variable passed from the RowCount transformation.

Question: Identify specific ETL scenarios that would call for the use of a user variable.

Demonstration: Adding a Variable to a Project

In this demonstration, you will learn how to:

- Create a variable
- Add a data flow task
- Add a variable message
- Add the data source
- Update variable
- Add a file destination

Question: How will this variable work in the final project?

Lesson 3

Building and Running a Solution

- Configuring Project Properties
- Saving a Solution
- Running a Package

In this lesson, you will cover the preparation of a package for execution. You will learn how to build a package, where it can be stored, and how to run it.

A package can have extensive usefulness. Packages can be scheduled to run on a recurrent basis or be deployed to multiple servers.

Configuring Project Properties



Package properties

specify the output path for files generated by the build process



Deployment Utility properties

Where to save deployment utility
Whether to allow updates



Debugging properties

Configure properties like Optimized Mode and Debugging Mode

Key Points

Properties for the project should be set during design time, prior to deploying and running the solution. There are three pages of default design-time property values for an Integration Services project.

- A project's configuration properties define settings related to designing, deploying, and debugging a project.
- Access a project's configuration settings through the SSIS menu or through the control flow design surface.
- On the Debugging page, you can specify several options that apply when you test the package by running it in Business Intelligence Development Studio at design time.

Saving a Package



Create a project in SQL, file system or package store as necessary



Each package is saved to a .dtsx file

Key Points

- A completed package is saved to the msdb database, the package store, or to a file when completed.
- Additional files are saved to the package store.
- Can be saved programmatically as well as through the interface.

Running a Package

Ways to run a package in Developer Studio

- Set startup package and run the project
- Right-click and run a package in SSIS Designer
- Open a package in Studio and run it
- Right-click and run individual tasks and containers



Key Points

A package can be run with:

- The dtexec command line directly from a prompt.
- The Execute Package Utility (`dtexecui.exe`).
- A SQL Server Agent job.
- At design time in the Business Intelligence Development Studio.

Note that all of these invoke the `dtexec` command.

Demonstration: Running a Package

In this demonstration, you will learn how to:

- Create an SMTP connection manager
- Create an event handler
- Build and run the package

Question: You ran the solution from the user interface. How else could you have executed the package?

Lab: Implementing an Integration Services Solution

- Exercise 1: Creating an Integration Services Project
- Exercise 2: Implementing a Package
- Exercise 3: Building and Running an Integration Services Package

Logon information

Virtual machine	NY-SQL-01
User name	NY-SQL-01\Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Creating an Integration Services Project

Scenario

The Marketing department has requested a list of products from the AdventureWorks database. They want the list in the format of a comma-delimited text file. You also want to view the number of rows returned each time you generate the file. Create an SSIS project that will create the file.

In this exercise, you will perform the foundational tasks for creating an SSIS solution.

The main tasks for this exercise are as follows:

1. Create an Integration Services project.
2. Rename the package and the package object.

3. View the tools available for control and data flow.
4. View the project and solution files.

► **Task 1: Create an Integration Services project**

- Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
- Start **Server Management Studio** and connect to the NY-SQL-01 SQL server database engine.
- In Business Intelligence Studio, create a new project called **E:\Mod02\labfiles\SSIS_proj2**.

► **Task 2: Rename the package and the package object**

- Rename the default package and package object to **products.dtsx**.

► **Task 3: View the tools available for control flow and data flow**

- View the toolbox objects available for the control flow tab.
- View the toolbox objects available for the data flow tab.

► **Task 4: View the file structure where the project and solution files are stored**

- In Microsoft Windows® Explorer, view the **E:\Mod02\Labfiles\SSIS_sol2** folder, and review the subfolders and files.

Results: After this exercise, you should have created and saved an SSIS package project.

Exercise 2: Implementing a Package

Scenario

Now that you have created the basic package, you need to add the objects that will perform the ETL.

In this exercise, you will connect the package to the data source, view the data with a data source view, and add objects to the package to extract data. In addition, you will employ a variable to display the current row count as the package processes.

The main tasks for this exercise are as follows:

1. Create a data source.
2. Create a data source view.
3. Create a connection manager.
4. Create a variable to hold the product count.
5. Define the control flow that will support the data flow operation.
6. Add a script to display the product count.
7. Add an OLE DB source to extract the product data.
8. Add a transformation to update the ProductCount variable.
9. Add a flat file destination.
10. Create an SMTP connection manager.
11. Create an event handler.
12. Configure the event handler to send an e-mail message after the Products.txt file is created.
13. Add annotations to the Products package.

► Task 1: Create a data source

- Create a new data source connection to the AdventureWorks2008 database on NY-SQL-01.
- Name the data source **AdventureWorks2008_ds**.

► **Task 2: Create a data source view**

- Create a new data source view using **AdventureWorks2008.ds** as the data source.
- Add the **Product (Production)** table to the **included** objects pane.
- Name the data source view **Products_dsv**.
- Expand the **Product** node in the **Tables** pane to view the columns in this table.
- In the Tables pane, right-click the **Product** node, and then click **Explore Data**.
- Review the sample data in the Table, PivotTable, Chart, and Pivot Chart tabs.

► **Task 3: Create a connection manager**

- From the control flow tab, create a new connection manager from the **AdventureWorks2008.ds** data source.
- Rename the new connection manager **AdventureWorks2008_cm**.

► **Task 4: Create a variable to hold the product count**

- On the **SSIS** menu, click **Add Variable**.
- Name the new variable **ProductCount**.
- Make sure that the variable matches the following criteria:
 - Scope: **Products**
 - Data type: **Int32**
 - Value: **0**

► **Task 5: Define the control flow to support a data flow operation**

- Add a **data flow** task to the control flow design surface.
- Name the new task **DFT Retrieve product data** and give it a description.

► **Task 6: Add a script to the control flow to display the product count in a message box**

- Drag a **Script Task** from the Toolbox to the control flow design surface.
- Click the **Data Flow** task, and then drag the precedence constraint from the Data Flow task to the Script task.
- Double-click the **Script** task.
- On the **General** page of the **Script Task Editor** dialog box, in the **Name** box, type **SCR Display product count**.
- In the **Description** box, type **Retrieve product count from ProductCount variable and display in message**.
- On the **Script** page of the **Script Task Editor** dialog box, in the **ReadOnlyVariables** box, type **ProductCount**, and then click **Edit Script**.
- In the **Microsoft Visual Studio for Applications** window, in the **Project Explorer pane**, double-click **ScriptMain.cs**.
- Select the **Main()** subroutine on the right hand drop down box. The **Main()** subroutine begins with the following line of code:

```
public void Main()
```

- Insert a line after **// TODO: Add your code here** and add the following code:

```
string MyVar;
MyVar = Dts.Variables["ProductCount"].Value.ToString();
MessageBox.Show(MyVar);
```

- Your **Main()** subroutine should now look like the following code:

```
Public Sub Main()
    string MyVar;
    MyVar = Dts.Variables["ProductCount"].Value.ToString();
    MessageBox.Show(MyVar);
    Dts.TaskResult = Dts.Results.Success
End Sub
```

- Press **ALT+Q** to close the **Microsoft Visual Studio for Applications** window.
- Click **OK** to close the **Script Task Editor** dialog box.
- On the toolbar, click the **Save All** button.

- ▶ **Task 7: Add an OLE DB Source to the data flow to extract the product data**
 - Open the data flow task design surface.
 - Drag an **OLE DB Source** from the Toolbox to the design surface.
 - Name the new object **OLE Retrieve product data** and provide a description.
 - On the **Connection Manager** page of the **OLE DB Source Editor** dialog box, add the **Products_dsv** data source view in the **OLE DB connection manager** list, using Table or view as the data source type.
 - Use the Production.product table.
- ▶ **Task 8: Add a Row Count transformation to the data flow to update the ProductCount variable**
 - Add a **Row Count transformation** to the design surface.
 - Connect the output of the **OLE DB source** to the new **Row Count transformation**.
 - Double-click the **Row Count** transformation.
 - In the **Advanced Editor for Row Count** dialog box, in the **Name** box, type **CNT Retrieve row count**.
 - In the **Description** box, type **Add row count to the ProductCount variable**.
 - In the **VariableName** box, type **ProductCount**.
 - Save the solution.

- ▶ **Task 9: Add a Flat File destination to the data flow to insert product data into the Products.txt file**
 - Add a **Flat File destination** to the design surface.
 - Connect the output of the **Row Count transformation** to the **Flat File destination**.
 - Name the flat file destination **FFD Load product data**.
 - In the **Flat File Destination Editor** dialog box, click **New**.
 - In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.
 - In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **Products_cm**.
 - In the **Description** box, type **Connects to the Products.txt file**.
 - In the **File name** box, type **E:\Mod02\Labfiles\\Products.txt**.
 - When completed, save all objects in the project.
- ▶ **Task 10: Create an SMTP connection manager**
 - Create a new connection manager of the type **SMTP** named **LocalSMTP_cm**.
 - Use **localhost** as the SMTP server.
 - Do not enable SSL or Windows Authentication.
- ▶ **Task 11: Create an event handler.**
 - Create an event handler for the **DFT Retrieve product data** node.
 - Use the **OnPostValidate** event handler type.

► **Task 12: Configure the event handler to send an e-mail message after the Products.txt file is created**

- Add a **Send Mail** task to the design surface.
- Name the task **SMTP send notification**.
- Use the **LocalSmtp_cm** connection for the **SMTPConnection**.
- In the **From** box, type **sqlserver@adventure-works.com**.
- In the **To** box, type **administrator@NY-SQL-01**.
- In the **Subject** box, type **Products updated**.
- In the **MessageSource** box, type **The Products.txt file has been updated**.
- Verify that **Direct Input** is selected in the **MessageSourceType** box and **Normal** is selected in the **Priority** box, and then click **OK**.

► **Task 13: Add annotations to the Products package**

- Create an annotation for the control flow. Type a description that provides an overview of the control flow.
- Create an annotation for the data flow. Type a description that provides an overview of the data flow.

Results: After this exercise, you should have added the objects necessary to create the transformation and creation of a flat file from the products table. The SSIS package will also be able to display the row count and send an email upon project completion.

Exercise 3: Building and Running an Integration Services Project

Scenario

In this exercise, you will complete the package and execute it, using debugging tools.

The main tasks for this exercise are as follows:

1. Build the Integration Services project.
2. Run the Integration Services project.
3. View the execution results.
4. Verify the package execution.

► **Task 1: Build the Integration Services Project**

- Save the package to the file system.
- Confirm that the project is saved to the E:\Mod02\Labfiles\SSIS_sol2\SSIS_proj2\bin folder.

► **Task 2: Run the Products package**

- Run the package from within Business Intelligence Studio.
- Compare the row counts associated with the data flow paths with the row counts displayed in the **Script Task** message box.

► **Task 3: Stop running the package and view the execution results**

- Click **Stop Debugging** on the toolbar.
- Click the **Execution Results** tab, and review the event information.

► **Task 4: Verify package execution**

- Verify that **Products.txt** was created and contains the products table data.
- Turn off virtual machine and delete changes.

Results: After this exercise, you should have run the package, creating the Products.txt file. The number of rows will have been displayed in a dialog box. Using the execution results tab, you will have identified problems with the SMTP server.

Module Review and Takeaways

- Review Questions
- Real-world Issues and Scenarios
- Best Practices

Review Questions

1. What tool will you most likely use to create an Integration Services package?
2. Besides the scripting example that you used today, identify another common use of variables in Integration Services projects.
3. What is the DTUtil command used for?

Real-world Issues and Scenarios

Several branch offices run identical database servers. A business analyst wants to collect the same set of information from all of these servers and collect them in an Excel spreadsheet. How would you develop and deploy a package that will perform this?

Best Practices Related to Integration Services Packages

Supplement or modify the following best practices for your own work situations:

- Create a naming convention for elements within the project.
- Annotate workspace.
- Debug solutions.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 3

Implementing Control Flow

Contents:

Lesson 1: Control Flow Tasks	3-3
Lesson 2: Control Flow Precedence Constraints	3-16
Lesson 3: Control Flow Containers	3-21
Lab: Implementing Control Flow	3-27

Module Overview

- Control Flow Tasks
- Control Flow Precedence Constraints
- Control Flow Containers

In this module, you will learn about the use of the control flow. The control flow in a SQL Server Integration Services (SSIS) package is constructed by using different types of control flow elements: the containers that provide structure in packages and services to tasks, tasks that provide functionality in packages, and precedence constraints that connect containers and tasks into a control flow.

You create the control flow in a package by using the control flow designer, the design surface on the Control Flow tab in SSIS Designer. Control flow directs the path of a solution from the collection of data to the insertion of data. It allows for a visual design of the solution.

Creating a control flow includes the following tasks:

- Adding containers that implement repeating workflows in a package or divide a control flow into subsets.
- Adding tasks that support data flow, prepare data, perform workflow and business intelligence functions, and implement script.
- Connecting containers and tasks into an ordered control flow by using precedence constraints.

Lesson 1

Control Flow Tasks

- Data Tasks
- Database Object Transfer Tasks
- Analysis Services Tasks
- File and Network Protocol Tasks
- Script and Program Execution Tasks
- Package Execution Tasks
- WMI Tasks
- Maintenance Plan Database Tasks
- Other Maintenance Plan Tasks

In this lesson, you will learn the major task objects that will make up most control flows. Using the SSIS Designer interface, you will see how tasks are assembled to constitute a package.

Data Tasks

Task	Definition
Data Flow task	<ul style="list-style-type: none">Runs data flows to extract data, apply column level transformations, and load data
Bulk Insert task	<ul style="list-style-type: none">Efficiently copies large amounts of data (text or BCP native format) into a SQL Server table or view
Execute SQL task	<ul style="list-style-type: none">Runs SQL statements or stored procedures from a package
Data Profiling task	<ul style="list-style-type: none">Identifies potential problems with data quality

Key Points

Use cases:

- Data Flow task** inserts a data flow into the project which can then be edited in the Data Flow tab.
- Bulk Insert task** can be used to copy large amounts of text or native database data into a SQL Server database as efficiently as possible.
- Execute SQL task** can be used to run any SQL command or stored procedure, such as a TRUNCATE command on a table.
- Data Profiling task** profiles data that is stored in SQL Server and to identify potential problems with data quality.

Database Object Transfer Tasks



Key Points

- Most server level objects have their own transfer task, including databases, error messages, jobs, and logins.
- The most often used task on this slide is the Transfer Database task.
- All transfer tasks can be performed between any Microsoft® SQL Server® 2000 or SQL Server database.
- System databases (master, msdb, tempdb) cannot be transferred.

Question: What would be the relative advantages between the Transfer Database task and the Bulk Insert task from the previous slide?

Demonstration: Data Flow and Object Transfer Tasks

In this demonstration, you will learn how to:

- Create a new Project
- Insert a Data Flow Task
- Insert a Transfer Login Task

Question: When would a Transfer Login task be useful?

Analysis Services Tasks

Task	Definition
Analysis Services Execute DDL task	<ul style="list-style-type: none">Runs data definition language (DDL) statements that can create, drop, or alter mining models and multidimensional objects such as cubes and dimensions
Analysis Services Processing task	<ul style="list-style-type: none">Processes Analysis Services objects such as cubes, dimensions, and mining models
Data Mining Query task	<ul style="list-style-type: none">Runs SQL statements or stored procedures from a package

Key Points

These tasks create, modify, delete, and process Analysis Services objects.

- The Execute DDL task is used for sending data definition language tasks to SQL Server Analysis Services.
- The Analysis Services Processing task can process multiple objects at the same time.
- The Data Mining Query task is used for predictive modeling, employing SQL Server Analysis Services.

Question: Do you utilize analysis services in your organization?

File and Network Protocol Tasks

Task	Definition
File System task	Creates, moves, or deletes directories and files
FTP task	Performs FTP upload and download actions
Message Queue task	Allows messages to transmit between packages or to custom applications
Send Mail task	Formats and sends an email message
Web Service task	Executes a Web service method
XML task	Retrieves, merges, validates, alters, and saves XML documents

Key Points

These tasks access data in various locations.

- Use the File System task to perform operations such as creating directories and moving files.
- The FTP tasks can deliver or collect files from an FTP site.
- Message Queue is used programmatically in conjunction with Microsoft® Message Queue Server. It is not used to send messages to operators.

Question: Do you utilize Web services that interact with your SQL database?

Demonstration: Adding a Send Mail Task

In this demonstration, you will learn how to:

- Create an SMTP connection manager
- Add a Send Mail task

Question: Besides notifying the user of package completion, what uses could the Send Mail task have? What if it utilized variables as well?

Script and Program Execution Tasks

Task	Definition
Script task	<ul style="list-style-type: none">Allows the execution of scripts for actions not available in other tasks
Execute Process task	<ul style="list-style-type: none">Runs an application or batch file
ActiveX Script task	<ul style="list-style-type: none">Legacy ActiveX capability. <i>Do not use for new development.</i>

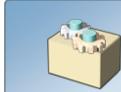
Key Points

These tasks execute external applications.

- The Script task uses Microsoft® Visual Studio® Tools for Applications (VSTA) as the environment in which you write the scripts and the engine that runs those scripts.
- Variables must be declared in advance in order to be passed in to or out of a script task.
- Scripts can be in either Microsoft® Visual Basic® 2008 or Visual C#® 2008.

Question: Are you currently utilizing any Microsoft® ActiveX® scripts?

Package Execution Tasks



Execute Package task

- Break up a complex workflow
- Reuse package parts
- Group work units
- Control security



Execute DTS 2000 Package task

- Use to employ legacy DTS packages created with SQL Server 2000 tools

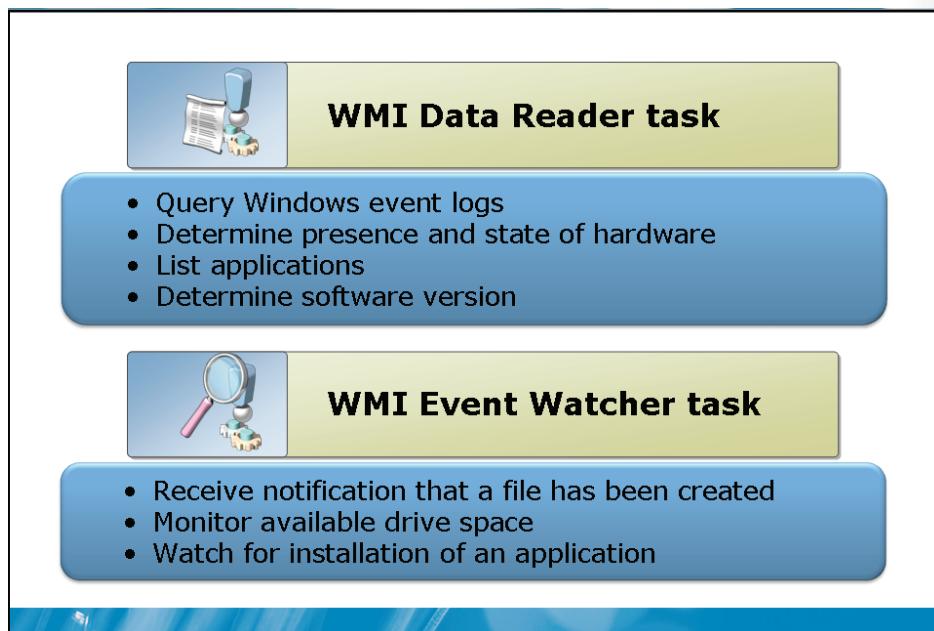
Key Points

Other SSIS packages can be called from within a package.

- Create modular, reusable packages and then use the Execute Package task to connect them.
- Execute DTS is purely a legacy tool for Microsoft® SQL Server® 2000 DTS packages.

Question: What types of activities could be put into a reusable package?

WMI Tasks



Key Points

- Windows Management Instrumentation (WMI) uses the Common Information Model (CIM) industry standard to represent systems, applications, networks, devices, and other managed components.
- Data Reader tasks runs queries at runtime. Use Data Reader tasks to identify hardware status and then decide whether to run another task.
- Event Watcher tasks will monitor a WMI property at runtime. Use Event Watcher tasks to check for the existence of files or monitor memory status.
- Provide access to event logs, installed software, and hardware configuration information among other WMI information.

Maintenance Plan Database Tasks

Task	Definition
Back Up Database	<ul style="list-style-type: none">Creates backups of multiple databases in full or simple model
Check Database Integrity	<ul style="list-style-type: none">Checks the allocation and structural integrity of all the objects in the specified database
Rebuild Index	<ul style="list-style-type: none">Rebuilds indexes in a single database or multiple databases
Reorganize Index	<ul style="list-style-type: none">Reorganizes indexes in a single database or multiple databases
Shrink Database	<ul style="list-style-type: none">Reduces the size of SQL Server database data and log files

Key Points

- Maintenance plans are packages created with the Maintenance plan wizard in Management Studio. Tasks are also available to the operator in SSIS Designer.
- Backup multiple or single databases using Back Up database tasks.
- Tasks utilize T-SQL statements, such as BACKUP.
- Use tasks such as Reorganize Index and Rebuild Index to perform regular maintenance on a database.

Other Maintenance Plan Tasks

Task	Definition
Execute SQL Server Agent Job	<ul style="list-style-type: none">Runs jobs that execute T-SQL statements and ActiveX scripts, perform Analysis Services and Replication maintenance tasks, or run packages or monitor SQL
Execute T-SQL Statement	<ul style="list-style-type: none">Executes a T-SQL statement, such as SELECT
History Cleanup	<ul style="list-style-type: none">Deletes entries in the msdb history tables
Notify Operator	<ul style="list-style-type: none">Sends notifications to SQL Server Agent operators
Update Statistics	<ul style="list-style-type: none">Updates information about the distribution of key values for one or more statistics groups (collections)

Key Points

These additional tasks are utilized in maintenance plans for interactions with other processes.

- SQL Server Agent Jobs can perform a range of activities, including scripts and replication tasks. Use the Execute SQL Server Agent Job task if you already have jobs that perform certain tasks.
- If you need to run parameterized queries, property expressions, or interact with variables, use the Execute SQL task instead of the Execute T-SQL task.
- Configure SQL Server operators, either in Management Studio or via T-SQL Scripts before using the Notify Operator task.

Lesson 2

Control Flow Precedence Constraints

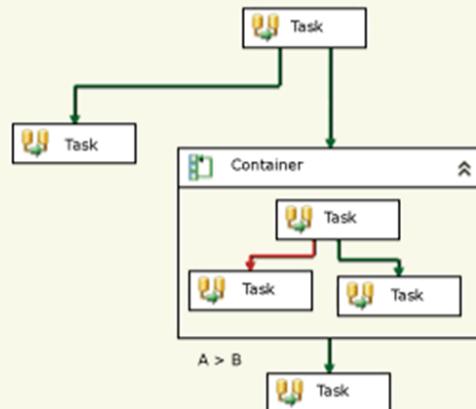
- Introduction to Precedence Constraints
- Execution Flow Based on Task Outcome
- Using Precedence Constraint Expressions

Precedence constraints connect tasks on the design surface. They provide the “Flow” of the control flow. Visually, the precedence constraints allow us to see how our package performs; the order tasks execute, and what types of conditional paths are in place.

Operationally, precedence constraints control the relationship between tasks at runtime. They allow you to create task flows which react to real situations.

Introduction to Precedence Constraints

Precedence constraints determine a package's execution flow



Key Points

During design time, precedence constraints are visualized as lines connecting tasks.

- A precedence constraint links two objects, known as the **precedence executable** and the **constrained executable**.
- If a control flow branches, the Integration Services run-time engine determines the execution order among the tasks and containers that immediately follow the branching.

Execution Flow Based on Task Outcome

Success	Precedence executable must run successfully for the constrained executable to run
Failure	Precedence executable must not succeed for the constrained executable to run
Completion	Constrained executable runs after the precedence executable runs

Key Points

A precedence constraint can be conditional to the outcome of the precedence executable.

- Use Failure for alternate paths or to notify an operator (possibly with a Send Mail task) of an error.
- Success is the default precedence constraint.

Question: When would you employ a completion precedence constraint?

Using Precedence Constraint Expressions

- Must be a valid Integration Services expression that evaluates to True or False
- Can be combined with constraint values
 - Expression only
 - Expression and constraint value
 - Expression or constraint value

Multiple constraints can be combined with AND/OR logic

The diagram illustrates a control flow starting from Task A. Two green arrows branch from Task A to Task B and Task C respectively. Above Task B is the text "Success and Condition 1". Below Task C is the text "Success and Condition 2".

```
graph LR; TA[Task A] -- "Success and Condition 1" --> TB[Task B]; TA -- "Success and Condition 2" --> TC[Task C]
```

Key Points

Use expressions with precedence constraints to determine path or execution in control flow or error handlers.

- Valid expressions include **expression grammar**, **expression evaluator**, and **expression builder**.
- Expressions are not only utilized in precedence constraints, but also in conditional split transformations, derived column transformations, and For Loop containers. They can also be used to set a variable value.
- Expressions can be used alone or used together with the execution result (success, failure, and completion).

Question: Describe a situation when an expression would be useful for a precedence constraint.

Demonstration: Introducing a Precedence Constraint

In this demonstration, you will learn how to:

- Connect the objects using a Success precedence constraint
- Run a package

Question: The operator needs to also be notified by email when a failure condition occurs, how would you implement that?

Lesson 3

Control Flow Containers

- Overview of Control Flow Containers
- Sequence Container
- For Loop Container
- Foreach Loop Container

In this lesson, you will introduce containers to the control flow. By implementing containers, you can loop a series of tasks, control the scope of a variable, or create more flexible precedence constraints.

Overview of Control Flow Containers



Containers provide a scope for defining variables



Containers are different from the grouping functionality



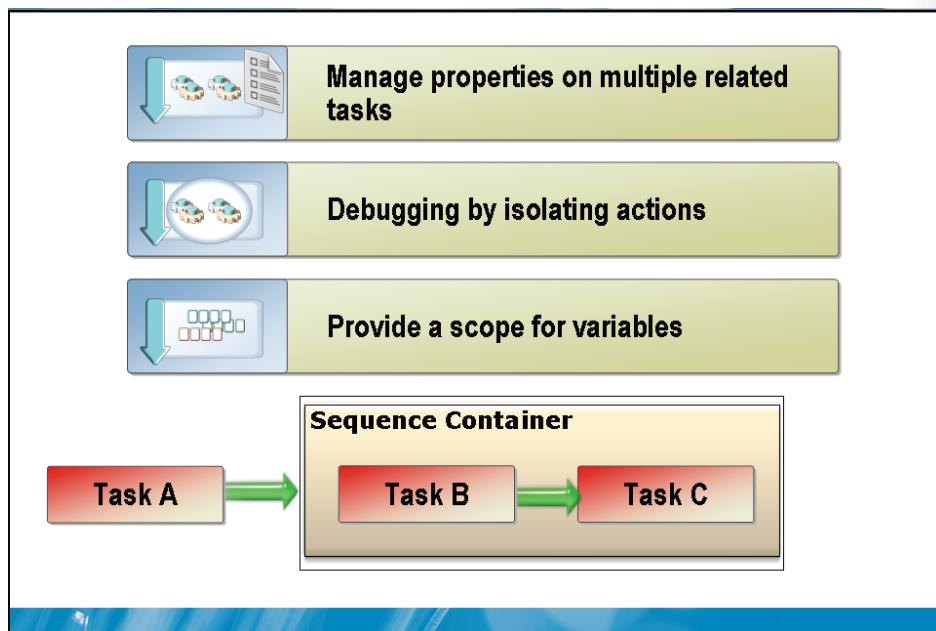
Containers provide scope for precedence constraints

Key Points

Containers hold multiple tasks and allow a consistent application of properties to them.

- **Foreach Loop Container** Runs a control flow repeatedly by using an enumerator.
- **For Loop Container** Runs a control flow repeatedly by testing a condition.
- **Sequence Container** Groups tasks and containers into control flows that are subsets of the package control flow.
- **Task Host Container** Provides services to a single task. All tasks are within a task host container.

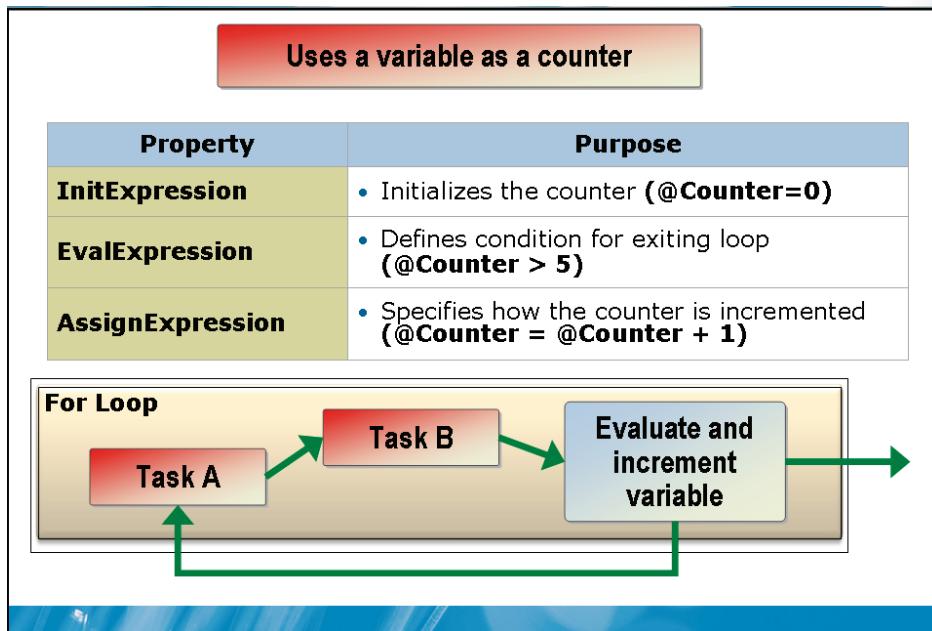
Sequence Container



Key Points

- Use sequence containers to organize objects during development.
- Useful to troubleshoot and test. Reroute precedence constraints to or from a single container when trying iterations.
- Divide package development into transactions.
- Set properties and variables at the container level.

For Loop Container



Key Points

Loops are defined with these three elements:

- **InitExpression** An optional initialization expression that assigns values to the loop counters.
- **EvalExpression** An evaluation expression that contains the expression used to test whether the loop should stop or continue.
- **AssignExpression** An optional iteration expression that increments or decrements the loop counter.

Foreach Loop Container

Foreach Enumerator	Use case
File	Perform tasks for all files in a folder with the .txt extension
Item	Enumerate the names of executables and working directories that an Execute Process task uses
ADO	Collect all rows in an ADO recordset
AD.NET	Enumerate the schema information about a data source, such as a list of tables
From Variable	Enumerate an array, an ADO.NET DataTable, an Integration Services enumerator,etc
NodeList	Enumerate the result set of an XML Path Language (XPath) expression
SMO	Act on SQL Management Objects, such as getting a list of the views in a database

Key Points

- Foreach loops are used to perform the same series of tasks on a number of items, such as files or tables.
- The File enumerator is used often in scenarios where a series of tasks need to be performed on all files in a given directory.
- Variables used in the Foreach Loop container can be utilized elsewhere.

Question: Besides file operations, when would you utilize a Foreach loop?

Demonstration: Creating a Foreach Loop Container

In this demonstration, you will learn how to:

- Create a variable
- Create a Foreach Loop container
- Execute the Project

Question: What system variables could we have added to the dialog box to expand the information available?

Lab: Implementing Control Flow

- Exercise 1: Creating a Simple Control Flow
- Exercise 2: Configuring Precedence Constraints
- Exercise 3: Using Containers

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Creating a Simple Control Flow

Scenario

After setting up the server, you need to create a new SSIS project. Create the connection managers and add the control flow tasks necessary to collect and distribute the data.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Create an Integration Services project.
3. Rename the package and the package object.
4. Create an OLE DB connection manager.
5. Create an SMTP connection manager.
6. Add an Execute SQL task to the control flow.
7. Add a Data Flow task to the control flow.

8. Add a Send Mail task to the control flow.
 9. Copy the Send Mail task.
- **Task 1: Set up the lab environment**
- Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
 - Browse to the E:\Mod03\Labfiles\Starter folder, and then double-click **Setup.cmd**. The script executes in a console window. Wait for the console window to close before proceeding.
- **Task 2: Create an Integration Services project**
- In Business Intelligence Development Studio, create a new project called **SSIS_proj3**.
 - Create the project in E:\Mod03\Labfiles\Starter.
 - Name the solution **SSIS_sol3**.
- **Task 3: Rename the package and the package object**
- Rename the package and package object **Employees.dtsx**.
- **Task 4: Create an OLE DB connection manager for the HumanResources database**
- Right-click the **Connection Managers** pane, and then click **New OLE DB Connection**.
 - Create a new connection to NY-SQL-01.
 - Use Windows Authentication for the connection.
 - Connect to the HumanResources database.
 - Test the connection.
 - In the **Connection Managers** pane, right-click the new connection manager, and then click **Rename**.
 - In the connection manager name box, type **HumanResources_cm**, and then press ENTER.

- ▶ **Task 5: Create an SMTP connection manager for the local server**
 - Right-click the **Connection Managers** pane, and then click **New Connection**.
 - In the **Add SSIS Connection Manager** dialog box, click **SMTP**, and then click **Add**.
 - In the **SMTP Connection Manager Editor** dialog box, in the **Name** box, type **LocalSmtp_cm**.
 - In the **Description** field, type **Connect to local SMTP server**.
 - In the **SMTP server** field, type **localhost**.
 - Verify that the **Use Windows Authentication** checkbox and the **Enable Secure Sockets Layer (SSL)** checkbox are cleared, and then click **OK**.
- ▶ **Task 6: Add an Execute SQL task to the control flow**
 - Add an **Execute SQL** task to the control flow.
 - Name the task **SQL Truncate Employees table**.
 - Use the **HumanResources_cm** connection manager and OLE DB for the connection type.
 - Use the following SQL Statement:

```
TRUNCATE TABLE hr.Employees
GO
```
 - Set **BypassPrepare** to **False**.
 - Save the project.
- ▶ **Task 7: Add a Data Flow task to the control flow**
 - Add a Data Flow task to the control flow
 - Name the task **DFT Retrieve employee data**.

► **Task 8: Add a Send Mail task to the control flow**

- Place a Send Mail task on the design surface.
- Name it **SMTP Send Successful Verification**.
- Configure it to use the **LocalSMTP_cm** connection manager.
- Create the following message:
 - From: sqlserver@adventure-works.com
 - To: administrator@adventure-works.com
 - Subject: Employee data loaded successfully
 - Message: The Employee package succeeded.
- Save the project.

► **Task 9: Copy the send mail task**

- Create a copy of the send mail task, naming the copy **SMTP Send failure notification**.
- Change the Subject to **Error loading data**.
- Change the message to **The Employee package failed**.

Results: After this exercise, you should have created the connection managers and tasks necessary for the Employees project.

Exercise 2: Configuring Precedence Constraints

Scenario

In this exercise, you will connect the tasks with precedence constraints and then be able to execute the package.

The main tasks for this exercise are as follows:

1. Connect a precedence constraint from the Execute SQL task to the Data Flow task.
 2. Connect precedent constraints from the Data Flow task to the Send Mail tasks.
 3. Test the control flow.
- **Task 1: Connect a precedence constraint from the Execute SQL task to the Data Flow task**
- Connect the output of the **Execute SQL** task to the input of the **Data Flow** task with a Success precedence constraint.
- **Task 2: Connect precedence constraints from the Data Flow task to the Send Mail tasks**
- Connect the output of **DFT Retrieve Employee Data** to **SMTP Send Success Notification** with a Success precedence constraint.
 - Connect the output of **DFT Retrieve Employee Data** to **SMTP Send Failure Notification** with a Failure precedence constraint.
- **Task 3: Test the control flow**
- Use debugging to execute the package.
 - Verify that the success path is completed without errors.

Results: After this exercise, you should have used precedence constraints to connect the objects of the package, then been able to execute the package.

Exercise 3: Using Containers

Scenario

In this exercise, you will introduce a Foreach loop container that will allow you to implement the package for any number of files.

The main tasks for this exercise are as follows:

1. Delete the precedence constraints.
2. Create a string variable to hold the file name.
3. Add a sequence container to the control flow and move the Execute SQL task to that container.
4. Add a Foreach Loop container to the Sequence container and move the Data Flow task into it.
5. Connect a precedence constraint from the Execute SQL task to the Foreach Loop container.
6. Connect precedence constraints from the Sequence container to the Send Mail tasks.
7. Add annotations to the control flow.
8. Run the Employees package.

► **Task 1: Delete the precedence constraints**

- Delete all of the precedence constraints created in Exercise 2.

► **Task 2: Create a string variable to hold the file name**

- Click the control flow design surface to ensure that no elements are active.
- From the SSIS menu, click **Variables**.
- In the Variables pane, click the **Add Variable** button.
- In the **Name** box, in the row for the new variable, type **EmployeeFile**.
- Verify that the **Scope** for the variable is set to **Employees**.
- In the **Data Type** list, click **String**.
- Click the **Close** icon to close the Variables pane.

- ▶ **Task 3: Add a Sequence container to the control flow and move the Execute SQL task to that container**
 - Add a sequence container to the control flow.
 - Name the new container **SEQ Process Employee data**.
 - Put the **SQL Truncate Employee table** task within the new container.
- ▶ **Task 4: Add a Foreach Loop container to the Sequence container and move the Data Flow task into the Foreach Loop container**
 - Add a Foreach Loop inside the **SEQ Process Employee** data container.
 - Rename it **FEL Enumerate employee files**.
 - Set the enumerator to **Foreach File Enumerator**.
 - Set the File enumerator to the **E:\Mod03\LabFiles\Starter\Employees** folder.
 - Perform the enumerator on all txt files.
 - Use the **User::EmployeeFile** variable.
 - Add the **DFT Retrieve Employee data** task to the new Foreach Loop container.
- ▶ **Task 5: Connect a precedence constraint from the Execute SQL task to the Foreach Loop container**
 - Add a success precedence constraint connecting the output of **SQL Truncate Employee table** to the Foreach loop container.
- ▶ **Task 6: Connect precedence constraints from the Sequence container to the Send Mail tasks**
 - Recreate the Send Mail precedence constraints from Exercise 2, Task 2, this time connecting the SEQ container with the send mail success and failure tasks.

► **Task 7: Add annotations to the task flow**

- Document the SMTP Send success notification, the SMTP Send failure notification with annotations.

► **Task 8: Run the Employees package**

- Run the Employees package using the debugger.
- Verify that all elements except the **SMTP Send Failure Notification** task function.
- Turn off the virtual machine and delete changes.

Results: After this exercise, you should have used containers to implement a loop in the package which allow the package to perform the same task on multiple files.

Module Review and Takeaways

- Review Questions
- Tools

Review Questions

1. What are the three possible precedence constraint outcomes?
2. What task would you employ to run an ALTER TABLE T-SQL command which utilizes a variable to specify what table to alter?
3. What container would you use to perform the same series of tasks on each column in a table?

Tools

Tool	Use for	Where to find it
Business Intelligence Design Studio	<ul style="list-style-type: none">Designing Control Flow	Installed with SQL Server 2008
Visual Studio Tools for Applications	<ul style="list-style-type: none">Creating and editing Scripts using Visual Basic 2008 and Visual C# 2008	Installed with SSIS

Module 4

Implementing Data Flow

Contents:

Lesson 1: Data Flow Sources and Destinations	4-3
Lesson 2: Basic Data Flow Transformations	4-11
Lesson 3: Advanced Data Flow Transformations	4-21
Lesson 4: Data Flow Paths	4-32
Lab: Implementing Data Flow	4-40

Module Overview

- Data Flow Sources and Destinations
- Basic Data Flow Transformations
- Advanced Data Flow Transformations
- Data Flow Paths

Microsoft® SQL Server® Integration Services provides three different types of data flow components: sources, transformations, and destinations. Sources extract data from data stores such as tables and views in relational databases, files, and Analysis Services databases. Transformations modify, summarize, and clean data. Destinations load data into data stores or create in-memory datasets.

Additionally, Integration Services provides paths that connect the output of one component to the input of another component. Paths define the sequence of components, and let you add annotations to the data flow on the designer surface or view the contents of the column.

This module begins with an overview of the types of data flow sources and destinations. It then discusses basic and advanced transformations in detail. It also covers data flow paths and how to implement data flow control.

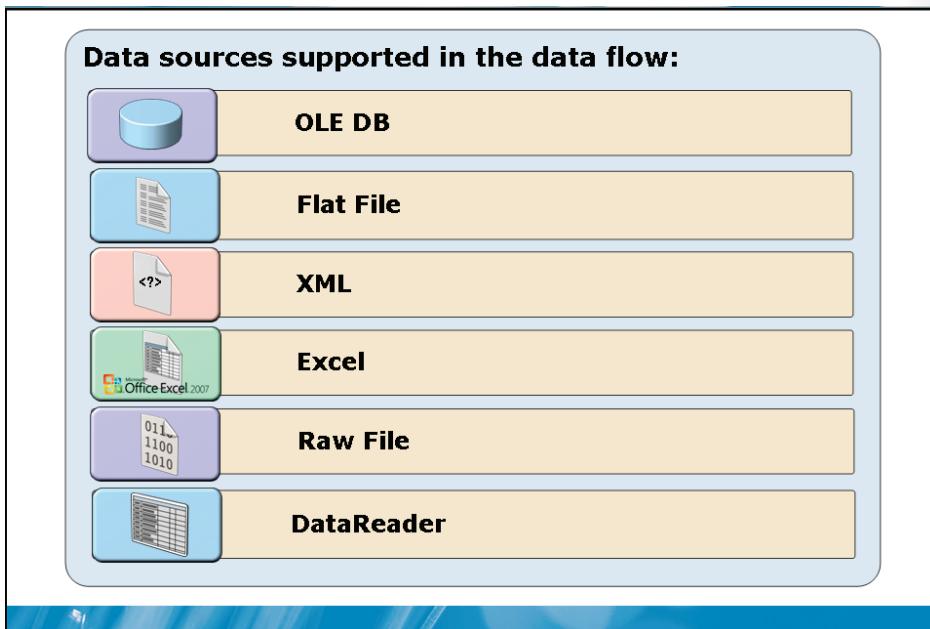
Lesson 1:

Data Flow Sources and Destinations

- Data Flow Sources
- Data Access Modes for OLE DB Data Sources
- Data Flow Destinations

In Integration Services, a source is the data flow component that makes data from different external data sources available to the other components in the data flow. A destination is the data flow component that writes the data from a data flow to a specific data store, or creates an in-memory dataset. In this lesson, you will learn about the different types of data sources and destinations and the options provided by Integration Services for accessing data sources and destinations.

Data Flow Sources



Key Points

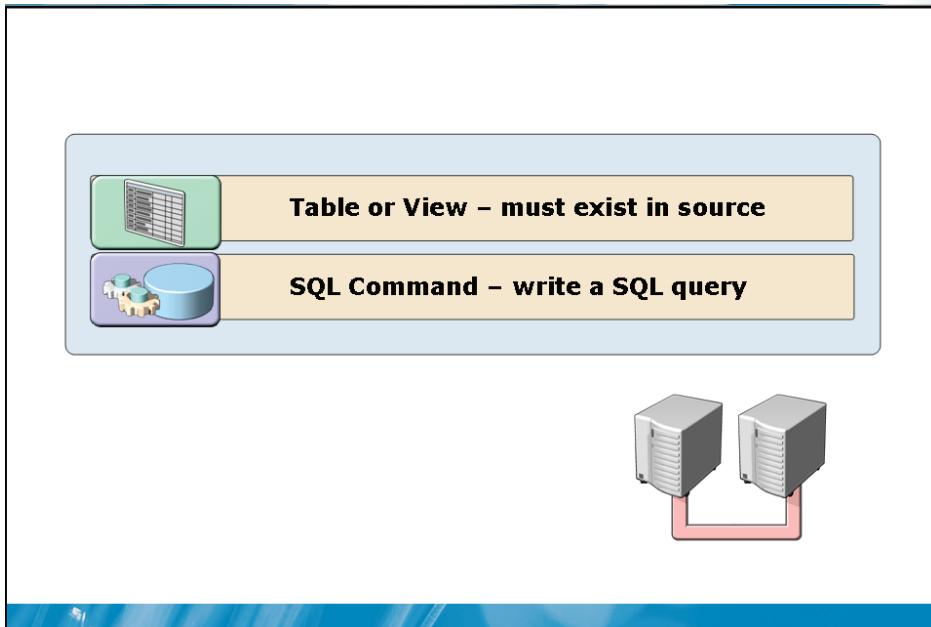
Source components access data stores and extract data. A data flow always begins with a source component. By using these components, you can access data from a wide variety of source types, including Microsoft SQL Server and Microsoft Office Access® databases, Microsoft Office Excel® worksheets, text files, and any source you can access through one of the .NET Framework data providers.

- The **OLE DB** source retrieves data from an OLE DB relational data source, such as SQL Server or Access. You connect to the data source through an OLE DB connection manager.
- The **Flat File** source retrieves data from a flat file, such as a text file, log file, or comma-separated values (CSV) file. You connect to the file through a flat file connection manager.
- The **XML** source retrieves data from an XML file. You connect to the file directly through the component, not through a connection manager. The XML source requires a schema to read the XML data. The schema can be an XML Schema Definition (XSD) file or an inline schema.

- The **Excel** source retrieves data from a Microsoft Excel file. You connect to the file through an Excel connection manager.
- The **Raw File** source retrieves data from a raw file. You connect to the file directly through the component, not through a connection manager. A raw file is a special type of file used by Integration Services for fast data reading and writing. The Raw File source reads data from a raw file that was written by a Raw File destination.
- The **DataReader** source retrieves data through a .NET provider that connects to a data source. You define an SQL statement to retrieve data from that data source.

Question: When might more than one data source apply to a scenario?

Data Access Modes for OLE DB Data Sources



Key Points

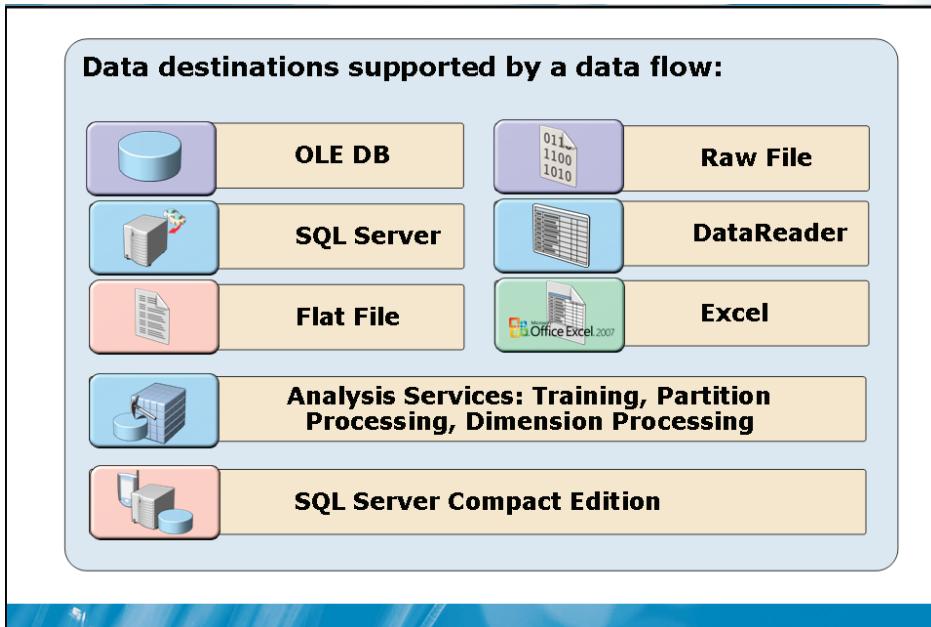
When you configure an OLE DB source or an Excel source, you must specify a data access mode. The data access mode refers to the method that the component will use to retrieve data.

- You can access data directly from a **table or view** in the data source. For the OLE DB source, the table or view must exist in the database specified in the connection manager. For an Excel source, the table or view refers to one of the worksheets in the Excel spreadsheet specified through the connection manager. For both components, you can specify the object name directly or through a variable.

- You can access data through a **SQL command**. This is the recommended method. If you specify this option, you can define the command directly in the component or reference the command through a variable. The SQL statement must access data from the database specified in the connection manager. One advantage to using an SQL statement is that you can include an ORDER BY clause to sort the data. By sorting the data when you extract it, you do not have to use a Sort transformation within the data flow.

Question: In your organization, which data access modes are currently implemented in one of your solutions?

Data Flow Destinations



Key Points

Destination components load data flow data into a target data store. A data flow ends with a destination component.

Destination	Description
OLE DB	Loads the data into an OLE DB relational data source, such as SQL Server or Access. You connect to the data source through an OLE DB connection manager.
SQL Server	Bulk-loads the data into a local SQL Server database. You connect to the data source through an OLE DB connection manager. You must also specify the target table of view. The component does not support any other data access modes.
Flat File	Loads the data into file such as a text file or CSV file. You connect to the file through a flat file connection manager. The format of the flat file can be delimited, fixed width, fixed width with row delimiter, or ragged right format.

(continued)

Destination	Description
Raw File	Loads the data into a raw file. You connect to the file directly through the component, not through a connection manager. The Raw File destination writes data to files that can later be used by a Raw File source in a separate operation. Raw files normally serve as an intermediary step of a larger process.
DataReader	Creates and populates an in-memory ADO.NET dataset based on the data. Other applications can then access the memory dataset.
Excel	Loads the data into an Excel worksheet. You connect to the Excel file through an Excel connection manager.
Analysis Services Data Flow Designations	<ul style="list-style-type: none">The Data Mining Model Training destination passes data through a training model algorithm to train a data mining model.The Partition Processing destination loads data into an Analysis Services partition and then processes that partition.The Dimension Processing destination loads data into an Analysis Services dimension and then processes that dimension.
SQL Server Compact Edition	Loads the data into a SQL Server Compact Edition database.

Demonstration: Implement a Data Flow

In this demonstration, you will see how to:

- Create a new data flow
- Run an SSIS package

Question: Why does the task named **Drop table(s) SQL Task** fail the first time you run the package?

Lesson 2:

Basic Data Flow Transformations

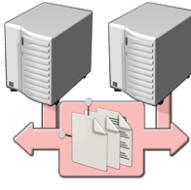
- Data Format Transformations
- Sorting and Grouping Transformations
- Column Transformations
- Lookup Transformations
- The Audit Transformation
- Custom Processing Transformations

SQL Server Integration Services transformations are the components in the data flow of a package that aggregate, merge, distribute, and modify data. Transformations can also perform lookup operations and generate sample datasets. This lesson describes the basic transformations that Integration Services includes and explains how they work.

Data Format Transformations

Data format transformations supported in a data flow:

	Character Map
	Data Conversion



Key Points

Data format transformations convert data as it passes through the data flow. By using these transformations, you can change data types, adjust value lengths, convert values to a different case, or perform a number of other operations.

- The **Character Map** transformation applies string operators to data as it passes through the data flow. For example, you can convert the values in a column to upper case.
- The **Data Conversion** transformation applies new data types and data type settings to data as it passes through the data flow. For example, you can convert values in a column from numeric types to strings.

Question: How might you use the Character Map transformation when working with data that will be displayed in a web page?

Sorting and Grouping Transformations

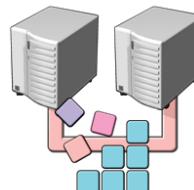
Sorting and Grouping transformations supported in a data flow:



Sort



Aggregate



Key Points

Sorting and grouping transformations reorganize data as it passes through the data flow. You can use these transformations to sort, aggregate, and group the data, as well as identify likely duplicates.

- The **Sort** transformation sorts data as it passes through the data flow. You can base the sorting on one or more columns, designate a sort order, and specify whether to sort the data in ascending order or descending order.
- The **Aggregate** transformation component applies aggregate functions to data as it passes through the data flow. The transformation can also group data, similar to a **GROUP BY** clause in a **SELECT** statement.

Question: Think of a scenario involving data for a product catalog. What types of sorting and grouping options would be appropriate?

Column Transformations

Column transformations supported in a data flow:



Copy Column



Derived Column



Export Column



Import Column

Key Points

Column transformations copy and create columns in the data flow. The transformations also enable you to import large files such as images or documents into the data flow or export the images or documents to a file.

- The **Copy Column** transformation creates an output column that is a duplicate of an input column. For example, you might want to copy a column so that you can convert the new column in some way, such as applying a Character Map transformation.
- The **Derived Column** transformation creates an output column based on an expression that incorporates one or more input columns. For example, you can create a derived column that concatenates two columns, one that contains first names and one that contains last names.

- The **Export Column** transformation exports text or data, such as an image in a column to a file. You can use the transformation to export images and documents to files.
- The **Import Column** transformation component imports text or data from a file into a column in the data flow. You can use the transformation to import images and documents into the data flow.

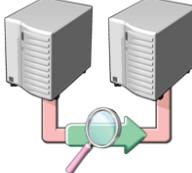
Question: The legal department wants to apply a copyright watermark to all product images in the database using a program that will run against a directory of image files. Which transformations might you use to help enable the process?

Lookup Transformations

Using Lookup transformations in a data flow:


Lookup data in another dataset


Outputs: Match, No Match, Error



Key Points

The **Lookup** transformation performs lookups by comparing data in an input column to data in a column retrieved from an OLE DB data source. The transformation joins the input data with the source data to perform the lookup. For example, suppose that your data flow includes the employee IDs, but not names. You can use the Lookup transformation to retrieve the names from the lookup table, based on the employee ID.

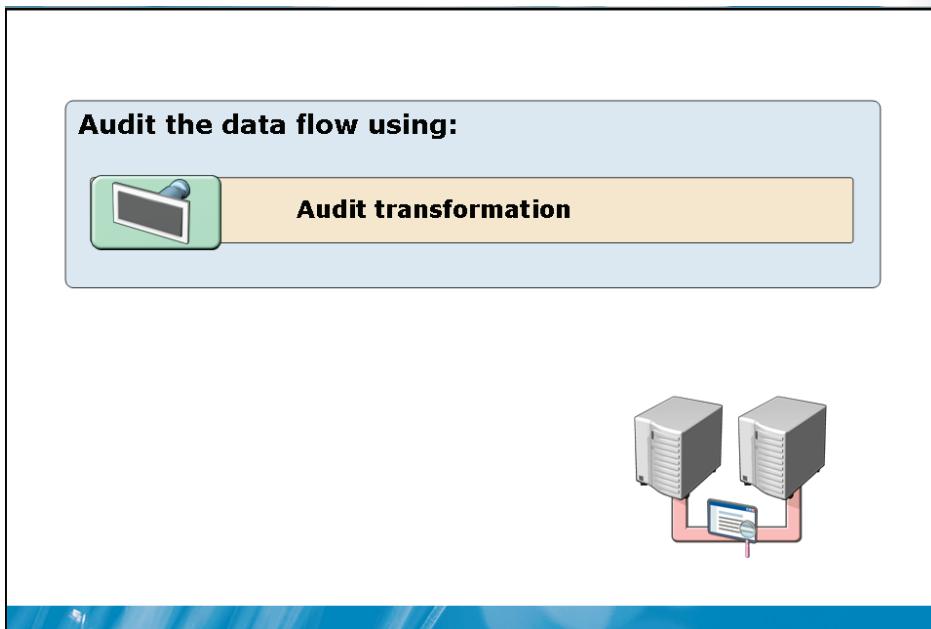
In SQL Server 2008, the Lookup transformation has three possible outputs:

- The **match** output handles the rows in the transformation input that do not match at least one entry in the reference dataset.
- The **no match** output handles rows in the input that do not match at least one entry in the reference dataset. If you configure the Lookup transformation to treat the rows without matching entries as errors, the rows are redirected to the error output. Otherwise, the transformation would redirect those rows to the no match output.

- The **error** output handles truncation errors and rows without matching entries if the component is configured to redirect them to the error output. (This is compatible with previous versions of SQL Server.)

Question: What advantages does the no match output provide over sending all rows without matches to the error output?

The Audit Transformation



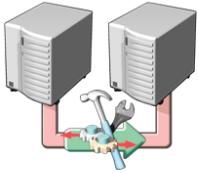
Key Points

The **Audit** transformation adds one or more columns to the data flow that provide information about the environment in which the package is running. The transformation uses system variables to provide the audit information; for example, you can add a column that tracks the execution start time and package name.

Custom Processing Transformations

Custom processing transformations available in a data flow:

	Script Component
	OLE DB Command



Key Points

- The **Script component** extends the data flow capabilities of Microsoft Integration Services packages with custom code written in Microsoft Visual Basic® 2008 or Microsoft Visual C#® 2008 that is compiled and executed at package run time. The Script component simplifies the development of a custom data flow source, transformation, or destination when the sources, transformations, and destinations included with Integration Services do not fully satisfy your requirements.
- The **OLE DB Command** transformation runs an SQL statement for each row in a data flow. For example, you can run an SQL statement that inserts, updates, or deletes rows in a database table.

Question: In what scenarios might you choose to use a script component data transformation?

Demonstration: Implement a Script Component Transformation

In this demonstration, you will see how to:

- Apply a custom transformation using the script component

Lesson 3:

Advanced Data Flow Transformations

- Fuzzy Transformations
- Split, Merge, and Join Transformations
- Data Sampling Transformations
- Data Analysis Transformations
- Term Transformations

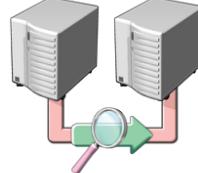
Integration Services provides advanced transformations that perform business intelligence and data distribution operations such as cleaning data, distributing and sampling data, running data mining prediction queries, and mining text. In this lesson, you will learn about these transformations and how to implement them in a data flow.

Fuzzy Transformations

Fuzzy transformations available in a data flow:

**Fuzzy Lookup**

**Fuzzy Grouping**



Key Points

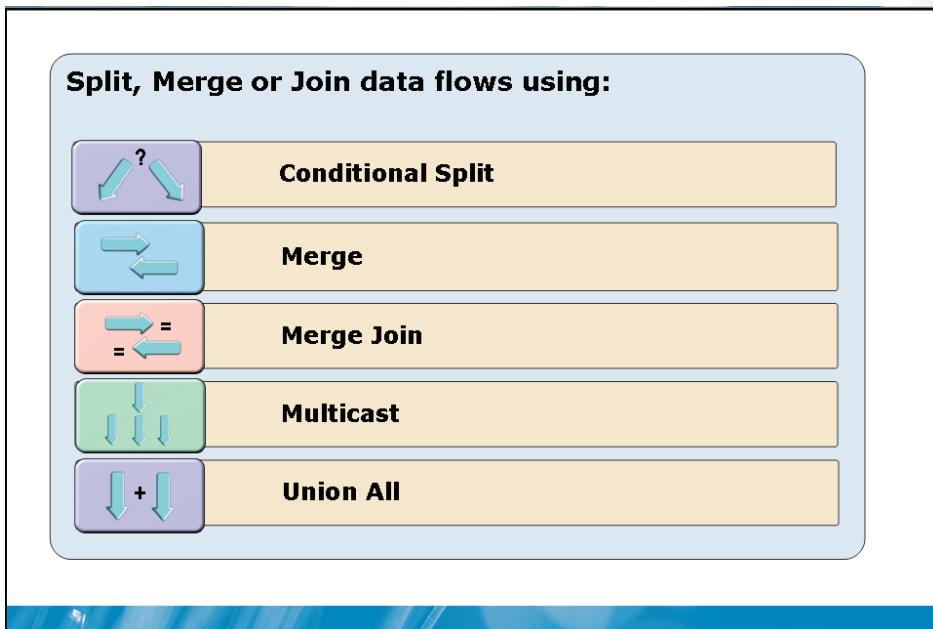
These SSIS transformations, available in SQL Server 2008 Enterprise Edition, are useful for improving the data quality of existing data as well as new data that is being loaded into your database.

- The **Fuzzy Lookup** transformation is similar to the Lookup transformation except that the Fuzzy Lookup transformation supports fuzzy matching. For example, you might want to use the transformation to look up names that are not an exact match. You also specify a similarity threshold that indicates how strictly the transformation identifies matches.

- The **Fuzzy Grouping** transformation component identifies likely duplicates in the data and provides mechanisms for standardizing the data. For example, you can use the transformation to group people with similar names to catch names that might be misspelled. When you configure the transformation, you specify a connection manager to a source that can be used to store temporary tables. Like the Fuzzy Lookup transformation, the Fuzzy Grouping transformation enables you to specify a similarity threshold that indicates how strictly the transformation identifies duplicates.

Question: How might you use the fuzzy lookup and fuzzy grouping options when dealing with data that is duplicated or not found due to misspellings?

Split, Merge and Join Transformations



Key Points

The split, merge, and join transformations provide a number of methods for creating multiple data flows or joining data flows.

Transformation	Description
Conditional Split	Divides rows based on the values within those rows. For each new data flow you want to create, you define an expression that limits the rows based on the specified criteria. For example, if your data flow includes employee information, you can split the data flow according to the regions in which the employees work.
Merge	Combines two data flows into a single data flow. The data flows must be sorted before they are merged.

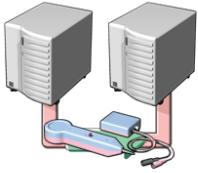
(continued)

Transformation	Description
Merge Join	Performs a join on two data flows to combine them into a single data flow. You can use the transformation to perform a full, left, or inner join. For example, you can perform an inner join on two data flows that extract data from different tables. By using an inner join, your output will include only matching rows. The data flows must be sorted before they are merged.
Multicast	Copies the data flow to create one or more additional data flows. The transformation is useful if you need to copy data to more than one data store or you want to perform different types of transformations on one of the data flows. For example, you might want to aggregate one of the data flows.
Union All	Similar to the Merge transformation, but the Union All transformation provides more flexibility. For example, if you use the Union All transformation for more than two inputs, you do not have to sort the inputs.

Data Sampling Transformations

Transformations that can be used to sample data:

	Percentage Sampling
	Row Sampling
	Row Count Transformation



© 2008 Microsoft Corporation. All rights reserved.

Key Points

The data sampling transformations are useful when you want to extract sample data from the data flow or you want to count the number of rows in the data flow.

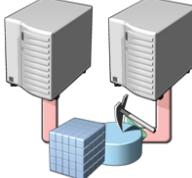
- The **Percentage Sampling** transformation divides the data flow into two data flows. The first data flow is based on the sample percentage of records that you specify when you configure the transformation. The second data flow contains the remaining records. The transformation is useful when you are developing a package that extracts a large amount of data. By sampling a percentage of data, you can reduce the size of your data flow while still processing a reasonable representation of the data. The transformation is also useful in data mining. You can use one output to train the data mining model and one output to test the model.

- The **Row Sampling** transformation is similar to the Percentage Sampling transformation, except that you specify the number of rows to sample, rather than the percentage.
- The **Row Count** transformation counts the rows in the data flow and stores that number in a variable. You can then access the variable from other elements in your package. For example, you can use the variable in an Execute SQL task to log the row count to a table.

Data Analysis Transformations

Data Analysis transformations available in a data flow:

	Pivot and Unpivot
	Data Mining Query
	Slowly Changing Dimension



Key Points

The data analysis transformations enable you to pivot and unpivot data and to perform tasks that support data warehousing and data mining.

- The data flow provides two pivot transformations for pivoting and unpivoting the data flow:
 - The **Pivot** transformation pivots the data flow on a column value. This process denormalizes the data and makes the data more compact. For example, your data flow might include sales data. Each row includes the salesperson, product, and amount of the sale. As a result, there are multiple rows for each salesperson. If you pivot the data set on the product column, the data set will now include one row for each salesperson and one column for each product. However, because not all salespeople sold all products, the data set will include many null values.

- The **Unpivot** transformation pivots the data set to make it more normalized. The impact of this transformation is the opposite of the Pivot transformation. For example, if you take the pivoted data set from the previous example, the Unpivot transformation would pivot the data set so that each row contained the salesperson, product, and amount of the sale.
- The **Data Mining Query** transformation performs Data Mining Extensions (DMX) prediction queries that evaluate the data flow data against a data mining model. You can use the transformation to run multiple prediction queries.
- The **Slowly Changing Dimension** transformation implements the steps necessary to transform and update the data in a slowly changing dimension in a data warehouse.

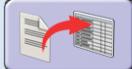
Question: Think of a database containing product pricing data. Which transformations might be used to create a table of suggested volume discounts based on historical sales data?

Term Transformations

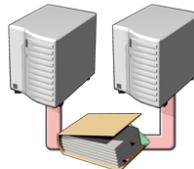
Term Transformations for manipulating text in a data flow:



Term Extraction



Term Lookup



Key Points

The term transformations compare words and phrases in your data flow to either a built-in English dictionary or to an OLE DB data source. You can use these transformations to search columns in your data flow for key words and phrases.

- The **Term Extraction** transformation component retrieves terms from an input column and writes those terms to an output column. The transformation uses English only, relies on its own dictionary, and extracts only nouns and noun phrases. For example, you can use the transformation to extract nouns and noun phrases from a column that stores customer e-mail messages to view the topics that the messages include.
- The **Term Lookup** transformation component is similar to the Term Extraction transformation, except that the Term Lookup transformation matches terms to a referenced table, rather than the built-in English dictionary. As a result, you can specify which terms to look up and from there determine how often these terms appear in the data flow column.

Demonstration: Implement a Conditional Split

In this demonstration, you will see how to:

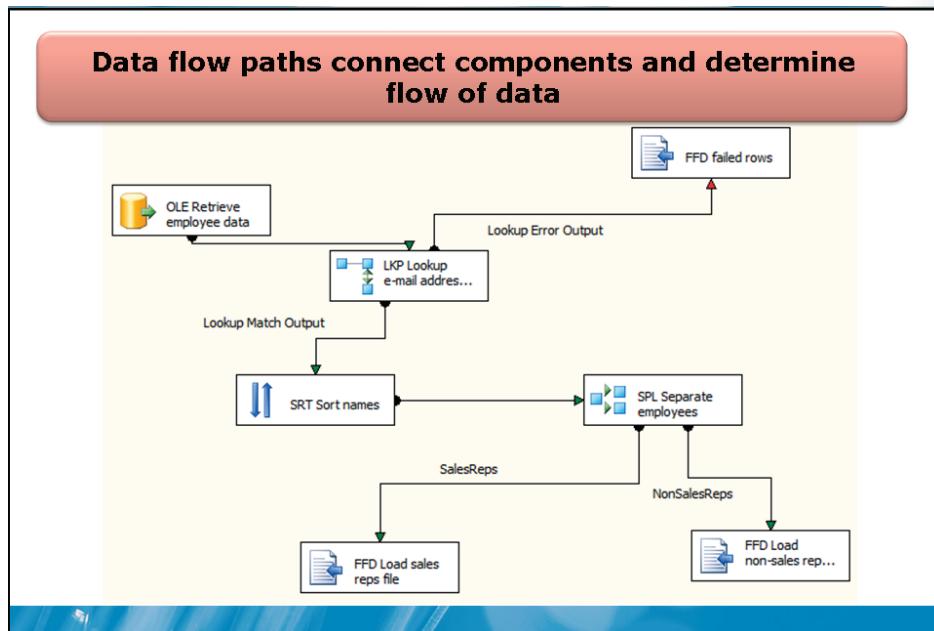
- Apply a Conditional Split to a Data Flow

Lesson 4: Data Flow Paths

- Introduction to Data Flow Paths
- What are Data Viewers?
- Using Data Viewers
- Handling Failed Rows

A path connects two components in a data flow by connecting the output of one data flow component to the input of another component. This lesson will discuss the data flow path, how to monitor the data flow using data viewers, and how to control the flow of failed rows.

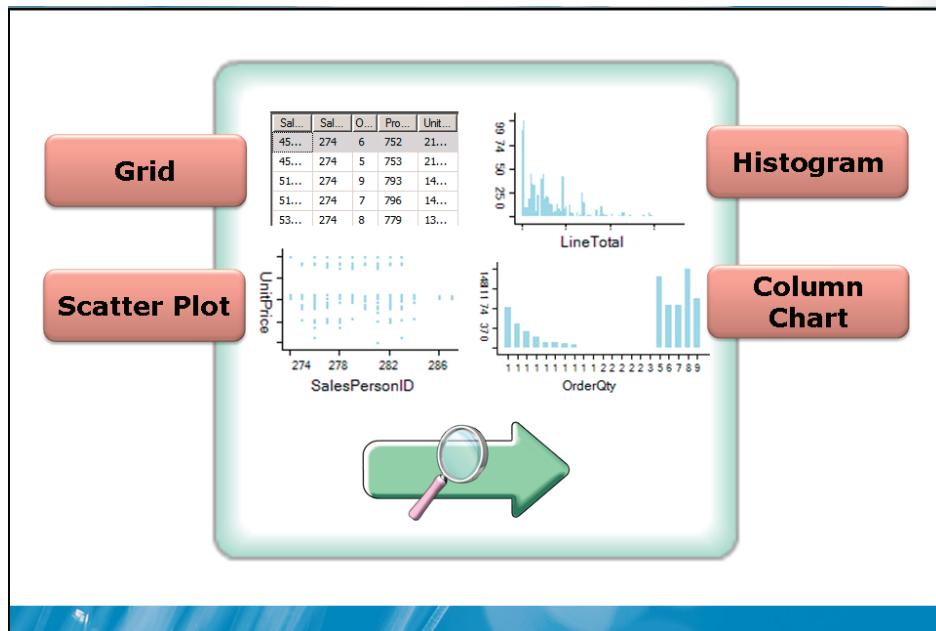
Introduction to Data Flow Paths



Key Points

- Data flow paths determine movement of data rows through the data flow components. A data flow path is green with an arrow at the downstream end of the path. Some components include an error output, which is red. You can redirect rows that raise an error, or might be truncated, to the error output.
- The output from one component becomes the input for the next component in the data flow. A component can have multiple inputs or outputs. Some components have only inputs or only outputs.
 - Sources have only outputs.
 - Transformations have inputs and outputs.
 - Destinations have only inputs.

What Are Data Viewers?



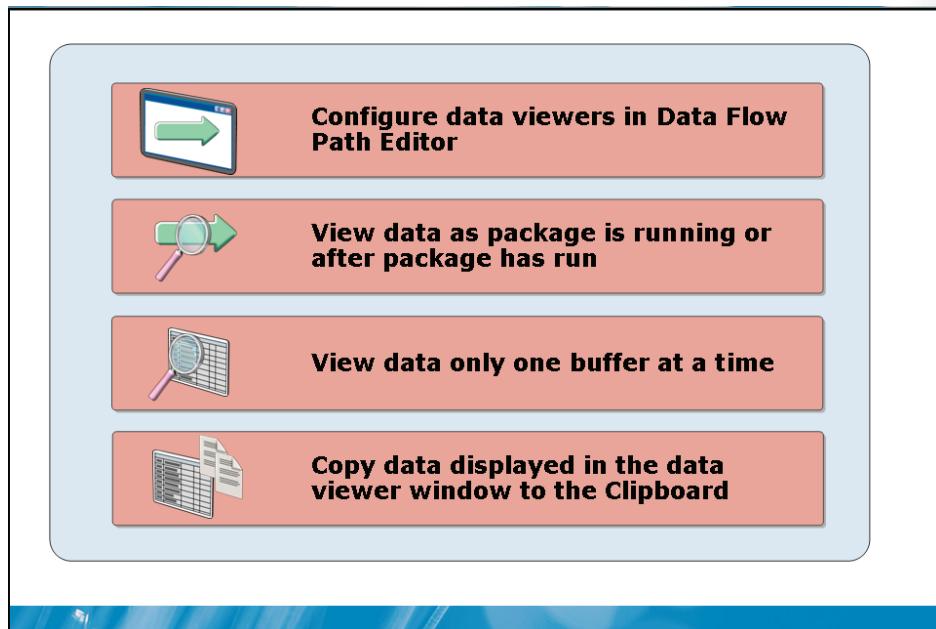
Key Points

A data viewer displays the data that is passing through a data flow path. For this reason, each data viewer is associated with a specific data flow path.

The data flow supports four types of viewers:

- **Grid.** This data viewer displays the data in a tabular format, like a table with columns and rows. You can select which columns to display in the data viewer.
- **Histogram.** This data viewer displays the distribution of numeric data in the format of a bar chart. You must select the column on which to model the data.
- **Scatter Plot.** This data viewer displays the distribution of numeric data on an X-axis and Y-axis. You must select the two columns (one for each axis) on which to model the data.
- **Column Chart.** This data viewer displays the count of each discrete value in a column. You must select the column on which to model the data.

Using Data Viewers



Key Points

You can add data viewers to any data flow path in your data flow. When data passes through a data flow path, a data viewer window opens for each data viewer that you have configured for that path.

- You configure data viewers in the **Data Flow Path Editor** dialog box. You can access the **Data Viewers** tab of the editor directly by right-clicking the data flow path and then clicking **Data Viewers**. Once in the editor, you can add and configure as many data viewers as necessary.
- You view data in data viewer windows as the package is running. When the data viewer window opens, the data flow stops running and waits for you to act. If you detach the window, the package restarts automatically. You can reattach the window at any time. You can also restart the package without detaching the window. The window remains open until you close it, whether or not it is detached, or until you stop running the package.

- A data viewer window shows data one buffer at a time. By default, the data flow pipeline limits buffers to about 10,000 rows. If the data flow extracts more than 10,000 rows, it will pass that data through the pipeline in multiple buffers. For example, if the data flow is extracting 25,000 rows, the first two buffers will contain about 10,000 rows, and the third buffer will contain about 5,000 rows. You can advance to the next buffer by clicking the green arrow in the data flow window.
- You can copy the data displayed in the data viewer window to the Clipboard. From there, you can add it to a file such as Microsoft Notepad or Excel.

Question: How can a data viewer help you debug your data flow?

Demonstration: Viewing Data in the Data Flow

In this demonstration, you will see how to:

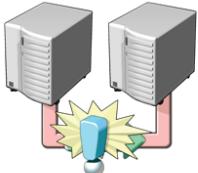
- Configure a data viewer
- View the data

Question: Why might you add a data viewer before and after an operation in the data flow path?

Handling Failed Rows

Handle a failed row in the data flow by:

	Specify action to take if an error or truncation occurs
	Fail component, ignore failure, or redirect row



Key Points

- For each column, you can specify the action to take if an error occurs or if a data truncation occurs. You can specify a different action for errors and truncations, or the same action for both. For example, if the value a data flow column is a string value and the component is expecting an integer, the component will treat the value as an error. However, if the component is expecting a string value, but the data flow value is too long, the component will treat that as a truncation.
- For each column, you can configure errors and truncations to take one of three actions:
 - **Fail component.** If the component encounters an incorrect value, the component will fail. This is the default setting for components that support error outputs.

- **Ignore failure.** If the component encounters an incorrect value, it will ignore that value and continue processing the data flow. In most cases, this will result in a null value being inserted into the data destination, unless a downstream component fails because of the value.
- **Redirect row.** If the component encounters an incorrect value, it redirects the row to the error output, which sends the row to a destination such as a flat file or a table in a SQL Server database.

Question: In what business scenarios might you use each action?

Lab: Implementing Data Flow

- Exercise 1: Transferring Data
- Exercise 2: Implementing Transformations
- Exercise 3: Using Data Viewers
- Exercise 4: Configuring Error Output

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Transferring Data

Scenario

The first file should contain sales data about orders for sales over \$10,000 per line item. The file should include the ID of the sales person who created the order.

This exercise's main tasks are:

1. Start the 6235A-NY-SQL-01 virtual machine and log on as Student.
2. Open the SSIS_sol4 solution.
3. Create a connection manager for the AdventureWorks2008 database.
4. Add an OLE DB Source to the DFT Create sales file data flow.
5. Add a Flat File Destination to the DFT Create sales file data flow.
6. Run the Employees package.
7. View the SalesData.csv file.

- ▶ **Task 1: Start the 6235A-NY-SQL-01 virtual machine and log on as Student, and set up the lab environment**
 - Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
 - Run the E:\MOD04\Labfiles\Starter\Setup.cmd file to create the **HumanResources** database and necessary folder structure.
- ▶ **Task 2: Open the SSIS_sol4 solution**
 - The **SSIS_sol4.sln** file is in the E:\MOD04\Labfiles\Starter\SSIS_sol4 folder.
 - On the **File** menu, click **New**, and then click **Project**.
 - When you have opened the solution, open the **Employees.dtsx** package in the **SSIS Designer**.
- ▶ **Task 3: Create a connection manager for the AdventureWorks2008 database**
 - Use the following settings for the connection manager:
 - Provider: **SQL Native Client 10.0**
 - Server name: **NY-SQL-01**
 - Authentication: **Windows Authentication**
 - Database: **AdventureWorks2008**
 - Test the connection.
 - Name the connection manager **AdventureWorks2008_cm**.
- ▶ **Task 4: Add an OLE DB Source to the DFT Create sales file data flow**
 - Use the following settings for the source:
 - Name: **OLE Retrieve sales data**
 - Description: **Retrieve data from SalesOrderHeader and SalesOrderDetail tables in AdventureWorks**
 - Connection manager: **AdventureWorks_cm**
 - Data access mode: **SQL command**

- Use the following SQL statement for the SQL command:

```
SELECT h.SalesOrderID, h.SalesPersonID,
       d.OrderQty, d.ProductID, d.UnitPrice,
       d.UnitPriceDiscount, d.LineTotal
  FROM Sales.SalesOrderHeader h JOIN
       Sales.SalesOrderDetail d
      ON h.SalesOrderID = d.SalesOrderID
 WHERE d.LineTotal > 10000
```

- Preview the data.
- **Task 5: Add a Flat File Destination to the DFT Create sales file data flow**
- Connect the data flow output from the OLE DB source to the Flat File destination.
 - Use the following settings for the Flat File destination:
 - Name: **FFD Load sales data**
 - Description: **Add data to the SalesData.csv file**
 - Connection manager: **create a new connection manager**
- Use the following settings for the new connection manager:
- Format: **delimited**
 - Name: **SalesData_cm**
 - Description: **Connects to the SalesData.csv file**
 - File name: **E:\MOD04\LabFiles\Starter\Employees\SalesData.csv**
 - Use the default column mappings.

► **Task 6: Run the Employees package**

- Run the package in debug mode.
- Verify that all components run successfully and that 766 rows pass through the DFT Create sales file data flow.

► **Task 7: View the SalesData.csv file**

- The file should be in the E:\MOD04\LabFiles\Starter\Employees folder.
- Verify that the file contains the sales data

Results: After this exercise, you should have successfully added a flat file destination and verified that the SalesData.csv file contains the correct data.

Exercise 2: Implementing Transformations

Scenario

You should also generate two text files based on employee data. Employee e-mail addresses are stored separately from employee details, so your data flow must include a lookup to retrieve e-mail addresses based on the employee ID. You should then sort the data by the employees' last names and then first names. The first text file should include only employees who are sales reps. The second text file should include all other employees.

This exercise's main tasks are:

1. Create a connection manager for the HumanResources database.
2. Add an OLE DB Source to the DFT Create employee files data flow.
3. Add a Lookup transformation to the DFT Create employee files data flow.
4. Add a Sort transformation to the DFT Create employee files data flow.
5. Add a Conditional Split transformation to the DFT Create employee files data flow.
6. Add a Flat File Destination for sales reps to the DFT Create employee files data flow.
7. Add a Flat File Destination for non-sales reps to the DFT Create employee files data flow.
8. Run the Employees package.
9. View the SalesReps.txt and NonSalesReps.txt files.

► **Task 1: Create a connection manager for the HumanResources database**

- Use the following settings for the connection manager:
 - Provider: **SQL Native Client 10.0**
 - Server name: **NY-SQL-01**
 - Authentication: **Windows Authentication**
 - Database: **HumanResources**
- Test the connection.
- Name the connection manager **HumanResources_cm**.

► **Task 2: Add an OLE DB Source to the DFT Create employee files data flow**

- Use the following settings for the source:
 - Name: **OLE Retrieve employee data**
 - Description: **Retrieve data from Employees table in HumanResources**
 - Connection manager: **HumanResources_cm**
 - Data access mode: **Table or view**
 - Table: **hr.Employees**
- Preview the data.
- Add an annotation to the data flow to describe the data extraction process.

- ▶ **Task 3: Add a Lookup transformation to the DFT Create employee files data flow**
 - Connect the data flow output from the OLE DB source to the Lookup transformation.
 - Use the following settings for the transformation:
 - Name: **LKP Look up e-mail addresses**
 - Description: **Retrieve e-mail address from EmailAddresses table**
 - Connection manager: **HumanResources_cm**
 - Table: **hr.EmailAddresses**
 - Joined column: **EmployeeID**
 - Lookup columns: **EmailAddress**
 - Lookup operation: **add as new column**
 - Add an annotation to the data flow to describe this transformation process.
- ▶ **Task 4: Add a Sort transformation to the DFT Create employee files data flow**
 - Connect the data flow output from the Lookup transformation to the Sort transformation.
 - Use the following settings for the transformation:
 - Name: **SRT Sort names**
 - Description: **Sort by last name, then first name**
 - Sort columns: **LastName, FirstName**
 - Sort order: **LastName, FirstName**
 - Sort type: **ascending**
 - Add an annotation to the data flow to describe this transformation process.

- ▶ **Task 5: Add a Conditional Split transformation to the DFT Create employee files data flow**
 - Connect the data flow output from the Sort transformation to the Conditional Split transformation.
 - Use the following settings for the transformation:
 - Name: **SPL Separate employees**
 - Description: **Separate sales reps from other employees**
 - Output names: **SalesReps, NonSalesReps**
 - SalesReps condition: **JobTitle == "Sales Representative"**
 - NonSalesReps condition: **JobTitle != "Sales Representative"**
 - Add an annotation to the data flow to describe this transformation process.
- ▶ **Task 6: Add a Flat File Destination for sales reps to the DFT Create employee files data flow**
 - Connect the data flow output for sales reps from the Conditional Split transformation to the Flat File destination.
 - Use the following settings for the Flat File destination:
 - Name: **FFD Load sales reps file**
 - Description: **Add data to the SalesReps.txt file**
 - Connection manager: **Create a new connection manager**
 - Use the default column mappings.
 - Use the following settings for the new connection manager:
 - Format: **delimited**
 - Name: **SalesReps_cm**
 - Description: **Connects to the SalesReps.txt file**
 - File name: **E:\MOD04\LabFiles\Starter\Employees\SalesReps.txt**
 - Add an annotation to the data flow to describe this transformation process.

► **Task 7: Add a Flat File Destination for non-sales reps to the DFT Create employee files data flow**

- Connect the data flow output for non-sales reps from the Conditional Split transformation to the Flat File destination.
- Use the following settings for the Flat File destination:
 - Name: **FFD Load non-sales reps file**
 - Description: **Add data to the NonSalesReps.txt file**
 - Connection manager: **Create a new connection manager**
 - Use the default column mappings.
 - Use the following settings for the new connection manager:
 - Format: **delimited**
 - Name: **NonSalesReps_cm**
 - Description: **Connects to the NonSalesReps.txt file**
 - File name: **E:\MOD04\LabFiles\Starter\Employees\NonSalesReps.txt**
- Add an annotation to the data flow to describe the data load process.

► **Task 8: Run the Employees package**

- Verify that all components run successfully and that 290 rows pass through the data flow—14 sales reps and 276 other employees.

► **Task 9: View the SalesReps.txt and NonSalesReps.txt files**

- The files should be in the E:\MOD04\LabFiles\Starter\Employees folder.
- Verify that the files contain the employee data.

Results: After this exercise, you should have successfully created a data flow with the proper transformations and verified that the SalesReps.txt and NonSalesReps.txt files contain the correct records.

Exercise 3: Using Data Viewers

Scenario

Now that you have created your data flows, you need to verify and test their functionality using data viewers.

This exercise's main tasks are:

1. Add a grid data viewer to the DFT Create sales file data flow.
2. Add a histogram data viewer.
3. Add a scatter plot data viewer.
4. Add a column chart data viewer.
5. Run the package, and view the data viewers.
6. Remove the data viewers.

► **Task 1: Add a grid data viewer to the DFT Create sales file data flow**

- Add the data viewer to the data flow path between the OLE DB source and the Flat File destination.
- Name the data viewer **Grid: sales data**.
- Include all columns except the **SalesPersonID** column.

► **Task 2: Add a histogram data viewer**

- Add the data viewer to the data flow path between the OLE DB source and the Flat File destination.
- Name the data viewer **Histogram: sales data**.
- Use the **OrderQty** column.

► **Task 3: Add a scatter plot data viewer**

- Add the data viewer to the data flow path between the OLE DB source and the Flat File destination.
- Name the data viewer **Scatter Plot: sales data**.
- Use the **OrderQty** column as the **X-axis** column.
- Use the **UnitPrice** column as the **Y-axis** column.

► **Task 4: Add a column chart data viewer**

- Add the data viewer to the data flow path between the OLE DB source and the Flat File destination.
- Name the data viewer **Column Chart: sales data**.
- Use the **OrderQty** column.

► **Task 5: Run the package, and view the data viewers**

- Run the package in debug mode.
- View the grid data viewer, and then detach it without closing it.
- View the other data views, and then close them.
- Close the grid data viewer after the package stops running.

► **Task 6: Remove the data viewers**

- Delete all of the data viewers from the data flow path.

Results: After this exercise, you should have successfully added a data viewer of each type, viewed the data in each data viewer, and then removed the data viewers from the data flow.

Exercise 4: Configuring Error Output

Scenario

Now, you need to configure your data flow to handle failed rows so that you can look at them later.

This exercise's main tasks are:

1. Add a row to the Employees table in the HumanResouces database.
2. Add a Flat File Destination for failed rows to the DFT Create employee files data flow.
3. Run the Employees package.
4. View the LookupErrors.txt file.

- ▶ **Task 1: Add a row to the Employees table in the HumanResouces database**
 - Run the E:\MOD04\Labfiles\Starter\AddEmployee.cmd file to insert a row into the hr.Employees table with no matching record in the hr.EmailAddresses table.
- ▶ **Task 2: Add a Flat File Destination for failed rows to the DFT Create employee files data flow**
 - Connect the error output from the Lookup transformation to the Flat File destination.
 - Configure the Lookup transformation to redirect rows for all errors and truncations.
 - Use the following settings for the Flat File destination:
 - Name: **FFD failed rows**
 - Description: **Add failed rows to the LookupErrors.txt file**
 - Connection manager: **Create a new connection manager**
 - Use the default column mappings.

- Use the following settings for the new connection manager:
 - Format: **delimited**
 - Name: **LookupErrors_cm**
 - Description: **Connects to the LookupErrors.txt file**
 - File name: E:\MOD04\LabFiles\Starter\Employees\LookupErrors.txt

► **Task 3: Run the Employees package**

- Verify that all components run successfully and that 291 rows pass through the data flow—1 error output row, 14 sales reps, and 276 other employees.

► **Task 4: View the LookupErrors.txt file**

- The file should be in the E:\MOD04\LabFiles\Starter\Employees folder.
- Verify that the file contains the error data for the employee with an **EmployeeID** value of **9999**.

Results: After this exercise, you should have successfully added a flat file destination for error output, run the package, and verified that the unmatched record appears in the LookupError.txt file.

Lab Shutdown

After you complete the lab, you must shut down the 6235A-NY-SQL-01 virtual machine and discard any changes.

Module Review and Takeaways

- Review Questions
- Best Practices

Review Questions

1. How does a data flow path relate to the data flow components?
2. Which transformations can be avoided by using a SQL Command as a data access mode?
3. Why not use SQL Server as a destination in a package?

Best Practices

Discuss the performance issues to avoid with transformations like aggregations.

Consider when sampling a data source would meet business requirements instead of copying the entire dataset. For example, in estimating average sales or certain data analysis and mining scenarios, using a small subset of data can produce statistically sound results without the burden of increased storage and performance.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 5

Implementing Logging

Contents:

Lesson 1: Overview of Integration Services Logging	5-3
Lesson 2: Using Logging	5-13
Lab: Implementing Logging	5-18

Module Overview

- Overview of Integration Services Logging
- Using Logging

Microsoft® SQL Server® 2008 Integration Services enables you to log detailed event information about a package's execution. You can configure Integration Services to log specific information about specific events, log specific information about all events, or log all information about all events. You can also specify whether to log that information on a specific task or container or at the package level. In addition, you can implement custom logging to meet logging requirements not supported by the built-in functionality.

This module begins with an overview of Integration Services logging, including providers and log elements. It then discusses configuring logging and how to implement logging a SSIS package.

Lesson 1:

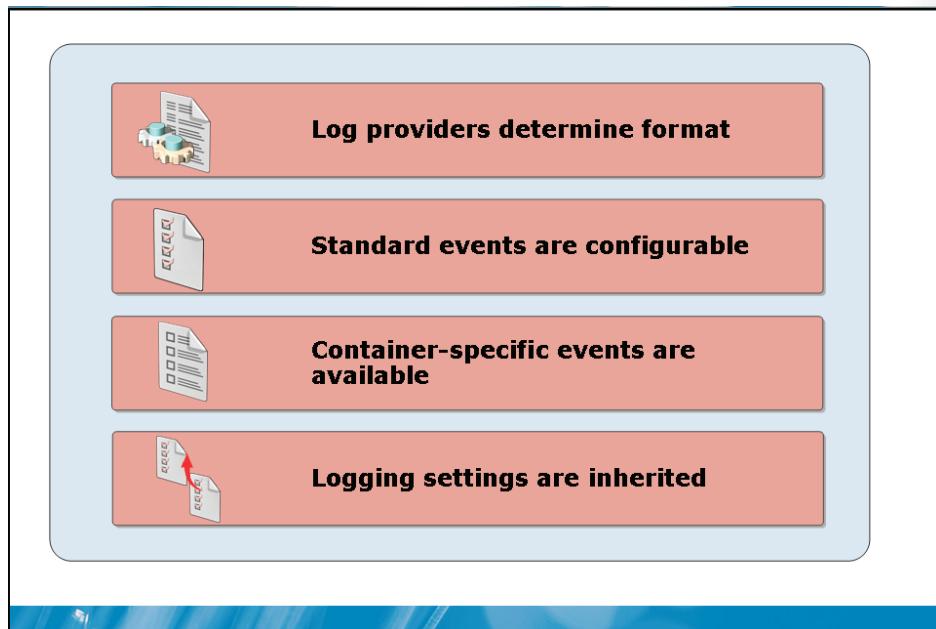
Overview of Integration Services Logging

- Introduction to Integration Services Logging
- Integration Services Log Schema
- Standard Diagnostic Log Entries
- Standard Execution Flow Log Entries
- Custom Log Entries
- Log Providers

Integration Services includes logging features that write log entries when run-time events occur and can also write custom messages. Integration Services supports a diverse set of log providers and gives you the ability to create custom log providers. The Integration Services log providers can write log entries to text files, SQL Server Profiler, SQL Server, Windows Event Log, or XML files.

In this lesson, you will learn how event logging is implemented and stored in Integration Services.

Introduction to Integration Services Logging



Key Points

Integration Services provides built-in features that enable you to log detailed event information about a package's execution. You can log different events on different executables or log the same events for all executables. You can also log the events to one data store or to multiple data stores, or specify which information to log for each event.

- Log providers determine the format of the log data as it is loaded into the target data store. You can specify one or more log providers for any container or executable in a package, and you can also specify different log providers for different containers or executables.
- Integration Services logging includes a set of standard events that you can configure on any container in a package.

- Individual container types support additional events. These custom events are specific to the container type.
- You can configure a container's logging to inherit the event and provider settings from its parent container, or you can prevent a child container from inheriting the parent settings.

Integration Services Log Schema

Element	Description
Computer	Name of the computer on which the log event occurred
Operator	Identity of the user who launched the package
SourceName	Name of the container or task in which the log event occurred
SourceID	Unique identifier of the package, container, or task in which the log event occurred
ExecutionID	The GUID of the package execution instance
MessageText	A message associated with the log entry
DataBytes	A byte array specific to the log entry. The meaning of this field varies by log entry

Key Points

When logging is enabled on a package, Integration Services writes an entry to the log for each event configured on the selected executable. The logged events adhere to a schema that defines the types of information included in each entry. By default, Integration Services includes all schema elements in an entry. However, you can override the default behavior and specify which elements to include.

Question: Why is the **ExecutionID** schema element important in a log file?

Standard Diagnostic Log Entries

 Element	 Description
OnError	Writes a log entry when an error occurs
OnWarning	Writes a log entry when a warning occurs
OnTaskFailed	Writes a log entry when a task fails
OnInformation	Writes a log entry during the validation and execution of an executable to report information
OnVariableValueChanged	Writes a log entry when the value of a variable changes
Diagnostic	Writes a log entry that provides diagnostic information

Key Points

Integration Services includes a set of standard events that you can enable on the package or on any container or task within the package. When a runtime event occurs, Integration Services writes an entry for the specific event to the log. The name of the event is the same as the name used in the log entry to identify the type of event.

Events	Description
OnError	Writes a log entry when an error occurs.
OnInformation	Writes a log entry during the validation and execution of an executable to report information.
OnTaskFailed	Writes a log entry when a task fails.

(continued)

Events	Description
OnVariableValueChanged	Writes a log entry when the value of a variable changes.
OnWarning	Writes a log entry when a warning occurs.
Diagnostic	Writes a log entry that provides diagnostic information.

Question: What are the advantages and disadvantages of logging all events versus only errors?

Standard Execution Flow Log Entries

Element	Description
OnPreExecute	Writes a log entry immediately before the executable runs
OnPreValidate	Writes a log entry when the validation of the executable starts
OnPostExecute	Writes a log entry immediately after the executable has finished running
OnPostValidate	Writes a log entry when the validation of the executable finishes
OnProgress	Writes a log entry when measurable progress is made by the executable
OnExecStatusChange	Writes a log entry when the execution status of the executable changes
OnQueryCancel	Writes a log entry at any juncture where it is feasible to cancel execution
PipelineComponentTime	For each data flow component, writes a log entry specifying processing time for each phase of validation and execution

Key Points

Integration Services includes entries relating to the execution flow of a package.

Question: Does your organization currently use SSIS logging? What events do you find in your logs?

Custom Log Entries

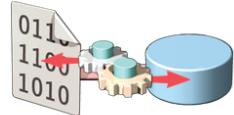
Many tasks provide custom events:



Specific events vary by executable



To log custom events, you must configure the executable explicitly



Key Points

Many executables support events that are specific to the type of executable. For example, a **Data Flow** task generates events specific to the data flow pipeline. You log these events in the same way as standard events.

- To log the custom events, you need to set logging options explicitly on the executable rather than inheriting log settings from the parent container.

Log Providers

Available log providers:

	Windows Event Log
	Text files
	SQL Server
	SQL Server Profiler
	XML files

Key Points

Log providers make it possible for Integration Services logging to write data to a data store. Log providers specify the format of the data.

- **SSIS log provider for Text files.** Writes to a text file. Requires a configured connection manager.
- **SSIS log provider for SQL Server Profiler.** Writes to a trace file that you can view in SQL Server Profiler. Requires a configured connection manager.
- **SSIS log provider for SQL Server.** Writes to the `sysdtslog90` table in a SQL Server database. Requires a configured connection manager.

- **SSIS log provider for Windows Event Log.** Writes to the Application log in the Windows Event Log.
- **SSIS log provider for XML files.** Writes to an XML file. Requires a configured connection manager.

Question: Why might you use the SQL Server Profiler log provider when logging?

Lesson 2: Using Logging

- Enabling Logging in a Package
- Configuring Logging for a Container or Component
- Implementing Custom Logging

To customize the logging of an event or custom message, Integration Services provides a schema of commonly logged information to include in log entries. The Integration Services log schema defines the information that you can log. You can select elements from the log schema for each log entry. Logs are associated with packages and are configured at the package level. Each task or container in a package can log information to any package log. You can select a level of logging that suits your needs by specifying the events to log and the information to log for each event.

In this lesson, you will learn how to implement built-in logging and custom logging in an Integration Services package.

Enabling Logging in a Package

To enable logging in a package:

- 1 Add one or more log providers**
- 2 Configure the log providers**
- 3 Select one or more executables**
- 4 Enable one or more providers on each executable**
- 5 Configure events on each executable**

Key Points

You configure Integration Services logging in the Configure SSIS Logs dialog box. To access the dialog box, click Logging on the SSIS menu. You can then add and configure data providers and select the executables to log and the events to record.

Configuring Logging for a Container or Component



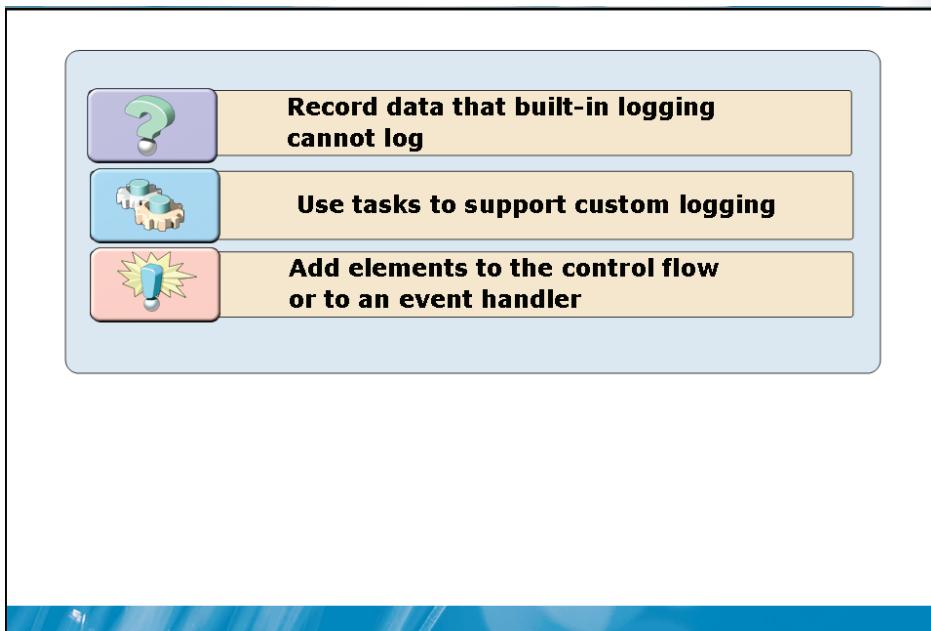
Key Points

After you have added and configured your log providers, you must select the executables and events you want to log. For each executable, you can use the check box in the Logging dialog box to log events, configure unique settings, or allow the executable to inherit its settings from a parent container.

- Clear the check box to prevent logging events on an executable.
- Select the executable check box to configure unique logging:
 - Load a configuration file to configure the events.
- Set the check box to gray to inherit the event settings from the parent container. If the check box is gray, no options on the Providers and Logs tab or the Details tab will be available. However, you can view the inherited settings.

Question: What advantages does using a configuration file provide?

Implementing Custom Logging



Key Points

In some cases, the built-in logging does not provide all your logging requirements for a particular package. Integration Services provides the functionality necessary to extend those logging capabilities.

- Use custom logging to record information that you cannot log through the default logging.
- Use tasks to implement the logic necessary to support custom logging.
- Add elements to the control flow or to an event handler to implement the logic necessary to support custom logging.

Question: Why might you use custom logging?

Demonstration: Implementing Logging

In this demonstration, you will see how to:

- Configure logging
- Implement logging within a package

Question: What is the first thing you must do to implement logging?

Lab: Implementing Logging

- Exercise 1: Configuring Logging
- Exercise 2: Implementing Custom Logging

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Configuring Logging

Scenario

You plan to log all error, warning, and information events generated by the data flow.

This exercise's main tasks are:

1. Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment.
2. Open the SSIS_sol5 solution in Business Intelligence Development Studio, and then open the Employees package.
3. Open the Configure SSIS Logs dialog box, and add the text file log provider.
4. Enable logging on the DFT Create employee files Data Flow task, and enable the text file log provider for that task.
5. Configure the events and information to be logged, and save the event configuration.

6. Enable logging on the DFT Create sales file Data Flow task, and enable the text file log provider for that task.
 7. Use the LogConfig.xml file to configure logging on the DFT Create sales file Data Flow task.
 8. Run the Employees package, and view the contents of the EmployeesLog.txt file.
- **Task 1: Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment**
- Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
 - Run the E:\MOD05\Labfiles\Starter\Setup.cmd file.
- **Task 2: Open the SSIS_sol5 solution in Business Intelligence Development Studio, and then open the Employees package**
- The SSIS_sol5.sln file is in the E:\MOD05\Labfiles\Starter\SSIS_sol5 folder.
 - When you have opened the solution, open the **Employees.dtsx** package in SSIS Designer.
- **Task 3: Open the Configure SSIS Logs dialog box, and add the text file log provider**
- To open the dialog box, click **Logging** on the **SSIS** menu.
 - Use the **SSIS log provider for Text files** provider type.
 - Use the following settings for the connection manager:
 - Usage type: **Create file**
 - File: E:\MOD05\Labfiles\Starter\Employees\EmployeesLog.txt

- ▶ **Task 4: Enable logging on the DFT Create employee files Data Flow task, and enable the text file log provider for that task**
 - Verify that the **Employees** checkbox in the **Containers** list is cleared.
 - You might need to click the **DFT Create employee files** checkbox once to clear the gray and then a second time to select it
- ▶ **Task 5: Configure the events and information to be logged, and save the event configuration**
 - Select the **OnError**, **OnInformation**, and **OnWarning** events.
 - Configure each event to return the following information:
 - **SourceName**
 - **SourceID**
 - **ExecutionID**
 - **MessageText**
 - Save the event configuration to **E:\MOD05\Labfiles\Starter\Employees\LogConfig.xml**.
- ▶ **Task 6: Enable logging on the DFT Create sales file Data Flow task, and enable the text file log provider for that task**
 - You might need to click the **DFT Create employee files** checkbox once to clear the gray, and then a second time to select it.
- ▶ **Task 7: Use the LogConfig.xml file to configure logging on the DFT Create sales file Data Flow task**
 - Click **Load** to access the LogConfig.xml file.
 - The file is in the **E:\MOD05\Labfiles\Starter\Employees\LogConfig.xml** folder.

► **Task 8: Run the Employees package, and view the contents of the EmployeesLog.txt file**

- Verify that all components run successfully.
- The log file should be in the E:\MOD05\LabFiles\Starter\Employees folder.
- Verify that the file contains the appropriate log data.

Results: After this exercise, you should have successfully configured the text file log provider, enabled logging, configured logging, run the package, and verified that the logged data appears in the EmployeesLog.txt file.

Exercise 2: Implementing Custom Logging.

Scenario

You also plan to log the number of rows into each destination data source every time the package runs.

This ‘exercise’s main tasks are:

1. Create a table in the HumanResources database to store the logged data.
2. Create variables to log the number of rows inserted into each data destination.
3. Add a Row Count transformation for sales rows to the DFT Create sales file data flow.
4. Add a Row Count transformation for sales rep rows to the DFT Create employee file data flow.
5. Add a Row Count transformation for non-sales rep rows to the DFT Create employee file data flow.
6. Create an event handler for the DFT Create sales file Data Flow task.
7. Add an Execute SQL Task to the event handler.
8. Create an event handler for the DFT Create employee files Data Flow task.
9. Add an Execute SQL task to the event handler.
10. Add a second Execute SQL task to the event handler.
11. Run the Employees package, and view the contents of the RowCounts table.

► **Task 1: Create a table in the HumanResources database to store the logged data**

- Run the Transact-SQL statement in the **RowCountsTable.sql** query file to create the log table.
- The file is located in the E:\MOD05\Labfiles\Starter\SetupFiles folder.
- You can use sqlcmd or SQL Server Management Studio to run the query.

- ▶ **Task 2: Create variables to log the number of rows inserted into each data destination**
 - Create one variable at the scope of the **DFT Create sales file** Data Flow task. Name the variable **SalesCount**.
 - Create two variables at the scope of the **DFT Create employee files** Data Flow task. Name the variables **RepsCount** and **NonRepsCount**.
 - All three variables should use the **Int32** data type and have an initial value of 0.
- ▶ **Task 3: Add a Row Count transformation for sales rows to the DFT Create sales file data flow**
 - Use the following settings for the transformation:
 - Name: **CNT Count sales rows**
 - Description: **Count rows and assign value to SalesCount**
 - Variable: **SalesCount**
- ▶ **Task 4: Add a Row Count transformation for sales rep rows to the DFT Create employee file data flow**
 - Use the following settings for the transformation:
 - Name: **CNT Count sales reps**
 - Description: **Count rows and assign value to RepsCount**
 - Variable: **RepsCount**
- ▶ **Task 5: Add a Row Count transformation for non-sales rep rows to the DFT Create employee file data flow**
 - Use the following settings for the transformation:
 - Name: **CNT Count non-sales reps**
 - Description: **Count rows and assign value to NonRepsCount**
 - Variable: **NonRepsCount**

► **Task 6: Create an event handler for the DFT Create sales file Data Flow task**

- Create an event handler for the **OnPostExecute** event of the **DFT Create sales file** executable.

► **Task 7: Add an Execute SQL Task to the event handler**

- Use the following settings for the Execute SQL task:
 - **Name:** SQL Insert sales log data
 - **Description :** Insert log data into RowCounts table
 - **Connection type:** OLE DB
 - **Connection manager:** HumanResources_cm
 - **SQL source type:** direct input
- Use the following INSERT statement to add a row to the **RowCounts** table:

```
INSERT INTO hr.RowCount
(ExecGUID, PackageName, SourceName,
DestinationName, NumberRows)
VALUES(?, ?, ?, 'FFD Load sales data', ?)
```

- Create four input parameter mappings for the following variables:
 - **ExecutionInstanceGUID** (data type: NVARCHAR; parameter name: **0**)
 - **PackageName** (data type: NVARCHAR; parameter name: **1**)
 - **SourceName** (data type: NVARCHAR; parameter name: **2**)
 - **SalesCount** (data type: LONG; parameter name: **3**)

► **Task 8: Create an event handler on the DFT Create employee files Data Flow task**

- Create an event handler for the **OnPostExecute** event of the **DFT Create employee files** executable.

► **Task 9: Add an Execute SQL task to the event handler**

- Use the following settings for the Execute SQL task:
 - Name: **SQL Insert sales reps data**
 - Description: **Insert log data into RowCounts table**
 - Connection type: **OLE DB**
 - Connection manager: **HumanResources_cm**
 - SQL source type: **direct input**
- Use the following INSERT statement to add a row to the **RowCounts** table:

```
INSERT INTO hr.RowCount
(ExecGUID, PackageName, SourceName,
DestinationName, NumberRows)
VALUES(?, ?, ?, ?, 'FFD Load reps file', ?)
```

- Create four input parameter mappings for the following variables:
 - **ExecutionInstanceGUID** (data type: NVARCHAR; parameter name: **0**)
 - **PackageName** (data type: NVARCHAR; parameter name: **1**)
 - **SourceName** (data type: NVARCHAR; parameter name: **2**)
 - **RepsCount** (data type: LONG; parameter name: **3**)

► **Task 10: Add a second Execute SQL task to the event handler**

- Connect a precedence constraint from the first Execute SQL task to the second Execute SQL task.
- Use the following settings for the second Execute SQL task:
 - Name: **SQL Insert non-sales reps data**
 - Description: **Insert log data into RowCounts table**
 - Connection type: **OLE DB**
 - Connection manager: **HumanResources_cm**
 - SQL source type: **direct input**

- Use the following INSERT statement to add a row to the **RowCounts** table:

```
INSERT INTO hr.RowCount
(ExecGUID, PackageName, SourceName,
DestinationName, NumberRows)
VALUES(?, ?, ?, ?, 'FFD Load non-reps file', ?)
```

- Create four input parameter mappings for the following variables:
 - **ExecutionInstanceGUID** (data type: NVARCHAR; parameter name: **0**)
 - **PackageName** (data type: NVARCHAR; parameter name: **1**)
 - **SourceName** (data type: NVARCHAR; parameter name: **2**)
 - **NonRepsCount** (data type: LONG; parameter name: **3**)
- **Task 11: Run the Employees package, and view the contents of the RowCounts table**
- Verify that all components run successfully.
 - View the contents of the **RowCounts** table in SQL Server Management Studio.
 - The **RowCounts** table should contain one row for each data destination.

Results: After this exercise, you should have successfully added variables, added Row Count destinations, created event handlers, added Execute SQL tasks, run the package, and verified that the row counts appear in the database table.

Lab Shutdown

After you complete the lab, you must shut down the 6235A-NY-SQL-01 virtual machine and discard any changes.

Module Review and Takeaways

- Review Questions
- Best Practices

Review Questions

1. When would you want to log to the Windows Event Log instead of other methods?
2. How do you configure custom events?
3. If you wanted to create real-time operation reports based on your SSIS logs, which logging provider should you use?
4. How many logging providers can you use for a package?

Best Practices

Use standard naming for each task or transform. For example, if you have a task that loads a sales table, you might name the Data Flow task “DFT Load Sales”. When you look at the resulting log, especially if using a tool to sort the events, the Source Name will indicate the task type.

Make log file names dynamic so that you get a new log file for each execution.

When debugging, always log to a text file, even if you have another log provider configured as well. A text file log has less dependencies and is most likely to contain all the information you need to debug a problem.

Use a template package to standardize your organization's logging and event handling.

To prevent log files from using large amounts of disk space, or to avoid excessive logging, which could degrade performance, you can limit logging by selecting specific events and information items to log. For example, you can configure a log to capture only the date and the computer name for each error.

Log data includes the name and the GUID of the package to which the log entries belong. If you create a new package by copying an existing package, the name and the GUID of the existing package are also copied. As a result, you may have two packages that have the same GUID and name, making it difficult to differentiate between the packages in the log data. To eliminate this ambiguity, you should update the name and the GUID of the new packages. In Business Intelligence Development Studio, you can regenerate the GUID in the ID property and update the value of the Name property in the Properties window. You can also change the GUID and the name programmatically, or by using the dtutil command prompt.

Module 6

Debugging and Error Handling

Contents:

Lesson 1: Debugging a Package	6-3
Lesson 2: Implementing Error Handling	6-13
Lab: Debugging and Error Handling	6-21

Module Overview

- Debugging a Package
- Implementing Error Handling

Integration Services provides numerous tools for debugging Integration Services packages and for handling errors. You can view a package's execution progress; use breakpoints, debug windows, row counts, and data viewers; and view log events. You can also implement event handlers that capture error information.

This module begins with an overview of Integration Services debugging tools, including breakpoints and debugging windows. It then discusses how to implement error handling in an SSIS package.

Lesson 1: Debugging a Package

- Viewing Execution Progress
- Using Breakpoints
- Using Debug Windows
- Using Row Counts and Data Viewers
- Debugging Scripts
- Viewing Log Events

Business Intelligence Development Studio and SSIS Designer include tools and features for debugging the control flow and data flow in a package, and for debugging the scripts in the Script task and Script Components.

In this lesson, you will learn how to set breakpoints on a package and its tasks and containers and then view information about the package in debug windows.

Viewing Execution Progress



View execution progress in control flow, data flow, and event handlers



Elements change color during the execution process



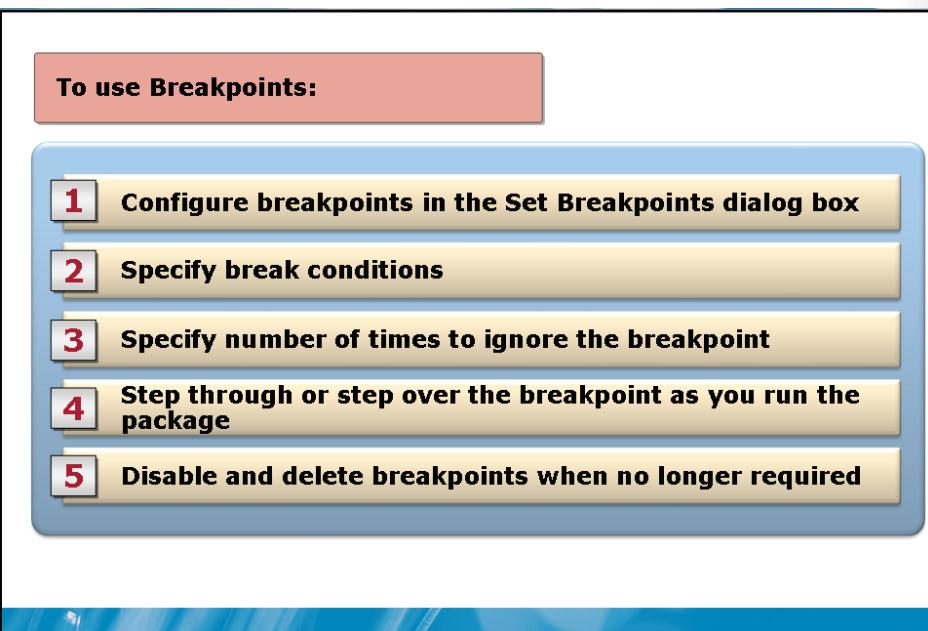
View execution progress on the Progress/Execution Results tab

Key Points

When you run a package in SSIS Designer, you can view the execution progress of that package. From this information, you can determine when a component begins to run and whether that component succeeds or fails. SSIS Designer also enables you to view event information during and after package execution.

- You can view the execution progress in the control flow, the data flow, and for any event handlers. If a package contains multiple Data Flow tasks, you can view the execution progress for each of those tasks.
- Elements in the control flow, data flow, and event handlers change color during the execution process.
- You can view the execution progress on the Progress tab.

Using Breakpoints



Key Points

SSIS Designer enables you to set breakpoints within a package. A breakpoint temporarily suspends the execution of a package so that you can verify different aspects of that execution. For example, you can use a breakpoint to examine the values assigned to variables at that point in the execution. You can also set additional breakpoints or disable those that already exist.

- You can set breakpoints on executables only, which include the package itself, as well as containers and tasks within that package. You can also set breakpoints on Microsoft® Visual Basic® .NET script defined in the **Script** task. In addition, SSIS Designer treats data viewers as breakpoints.
- You configure breakpoints in the **Set Breakpoints** dialog box.
- You select the break conditions that you want to associate with an executable.
 - You can specify the number of times Integration Services ignores a breakpoint before it triggers that breakpoint.

- You can step through or step over a breakpoint as you run the package.
- You can use one of several methods to disable and delete breakpoints.

Question: Why is a breakpoint necessary when debugging a package?

Using Debug Windows

Debug windows are different depending on the container or task currently being monitored:

- **Breakpoints window lists all package breakpoints**
- **Output window lists status messages**
- **Immediate window lists expressions and print variable values**



Key Points

SSIS Designer can display a variety of debugging windows when a package is running. You can use these windows to view information such as the output from debugging the package, the values assigned to variables, or a list of breakpoints configured on the package.

- The debug windows apply only to the package object and its containers and tasks, including containers or tasks in an event handler. The only exceptions to this are data viewers, which SSIS Designer treats as breakpoints even though they are part of the data flow.
- The debug windows include the **Breakpoints** window, which lists all breakpoints defined in your package.

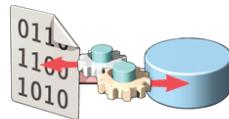
- The debug windows include the **Output** window, which lists the status messages generated during package execution.
- The debug windows include the **Immediate** window, which enables you to debug and evaluate expressions and print variable values.

Question: Why are there multiple debug windows?

Using Row Counts and Data Viewers

Row counts and data viewers apply only to the data flow :

- SSIS Designer displays number of rows that pass through pipeline
- Data viewers provide a snapshot of the data



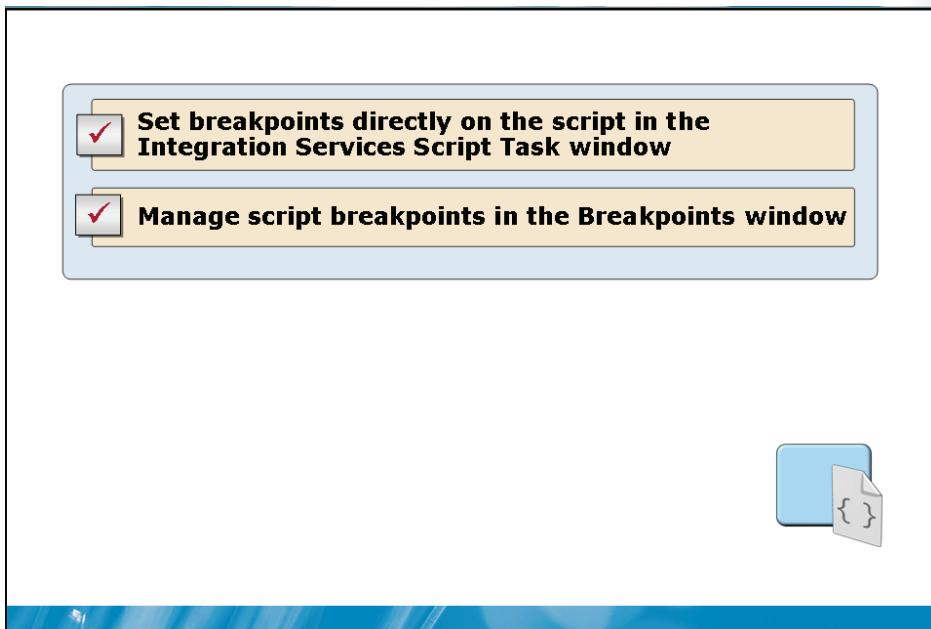
Key Points

When you debug a package, you can often find useful information by watching the data flow. When SSIS Designer processes the data flow, you can access row count information and use a data viewer to view data as it passes through the data flow.

- When you run a package, SSIS Designer displays the number of rows that pass through the data flow each time that data flow is executed.

Question: How might you use Row Count when debugging a data flow?

Debugging Scripts



Key Points

When you add Visual Studio .NET script to a **Script** task or Script component in a Data Flow, you can set breakpoints on individual lines of code. As with tasks and containers, the breakpoints stop the execution of the script. You can also view debugging windows that are specific to the script.

- You set breakpoints directly on the script in the editing window. You can set one breakpoint for each line of code.
- You can manage script breakpoints in the Breakpoints window. When you run a script, you can display a Breakpoints window specific to that script. The window displays only those breakpoints within the script. The script breakpoints also appear in the Breakpoints window available to the package. In either window, you can view which breakpoint is active and you can disable any of the breakpoints.

Demonstration: Debugging a Script

In this demonstration, you will see how to:

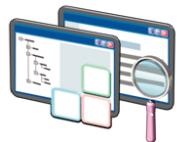
- Add breakpoints to a script

Question: How do the breakpoints in a script task work with the breakpoints set in the package?

Viewing Log Events

To open the Log Events window, click Log Events on the SSIS menu

- View events as Integration Services generates them at runtime
- Copy data from the Log Events window to the Clipboard



Key Points

The Log Events window provides an easy mechanism for viewing events generated during package execution. The window takes advantage of Integration Services logging. You can view events at runtime or after a package has executed. You can then copy the event data to a Clipboard to save to a file.

Question: Why is it important to implement best practices in your organization for logging as it applies to debugging?

Lesson 2:

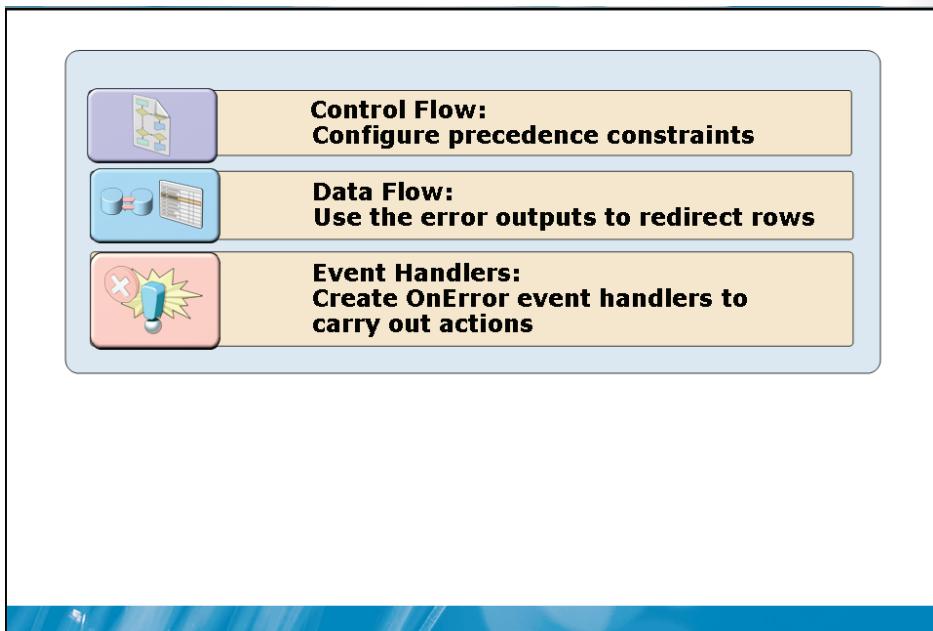
Implement Error Handling

- Overview of Error Handling
- Implementing Event Handlers
- Handling Script Errors
- Capturing Error Information
- Controlling Failure Behavior

Errors can occur in a package from a variety of sources. You can respond to the many events that are raised by the package and the objects in the package by using event handlers. You can also implement error handling in your scripts, and use custom logic to obtain system information for troubleshooting, or clean up temporary resources or incomplete output.

In this lesson, you will learn how to implement error handling in an SSIS package by using Event Handlers, using a Try/Catch block in a Script task, and controlling the failure behavior of tasks.

Overview of Error Handling



Key Points

When you create an Integration Services package, you should include in your design the logic necessary to handle errors that might occur when the package runs. Integration Services provides several features that you can use to enable error handling.

- Configure precedence constraints in the control flow to take a specific action in case a container or task fails, such as sending an e-mail message in case of failure.
- Use the error outputs in the data flow to redirect rows that would cause a component to fail.

- Create an **OnError** event handler to carry out actions when a container or task generates an error.

Question: What advantages does using an Event Handler give you when an error occurs?

Implementing Event Handlers



The OnError event handler is the most commonly used

Events propagate up through a package's executable hierarchy

Key Points

Package executables generate events as they are executed. You can create an event handler for any executable for any standard event. When the executable generates one of these events, Integration Services runs the event handler for that event.

- You can use the **OnError** event handler to take a specific action whenever an executable generates an **OnError** event.
- By default, events propagate up through a package's executable hierarchy. For example, if an error occurs in a Script task that is contained in a Sequence container, the **OnError** event for the Script task, the Sequence container, and the package itself will all fire.

Question: Why might you prevent an error from propagating?

Handling Script Errors



Use a Try-Catch block to handle exceptions

Use the Dts.Events.FireError method to fire an OnError event

Key Points

You can incorporate error handling into your Microsoft Visual Basic or Microsoft C#® script, and integrate it with your package's error handling. To incorporate error handling, use a **Try-Catch** block to catch exceptions, and then use the **FireError** method to fire an **OnError** event.

Question: Why is it important to raise an error in a script?

Capturing Error Information

Capture error information from system variables

- In a Script task, add variables as read-only and retrieve values from **Dts.Variables**
- In an Execute SQL task, map variables to parameters



Key Points

System variables store information about a package and its objects at run time. You can use these variables to capture data that will help you handle errors. You can then use the data in scripts and SQL statements to carry out any necessary action.

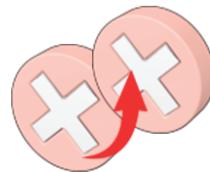
- To capture the system variable values in a script, add them to the Script task as read-only variables and retrieve their values from the **Dts.Variables** collection.

Question: What types of information might be found in system variables that would be of use in error handling?

Controlling Failure Behavior

By default, an error causes a container or task to fail

- You can configure the number of permitted errors
- You can control failure behavior on parent containers and package object
- You can configure an executable to fail
- You can set the DTS.TaskResult flag in script



Key Points

Integration Services supports a number of features that enable you to control the behavior of a package if an executable within that package fails.

- You can configure the **MaximumErrorCount** property on an executable to permit a specific number of failures.
- You can force an executable to fail by setting the **ForceExecutionResult** property. This technique can be useful when debugging or testing a package.
- You can set the **Dts.Results** flag in a script, to control the execution result of the Script task.

Question: In what scenario might you use a MaximumErrorCount?

Demonstration: Monitoring and Debugging an SSIS Package

In this demonstration, you will see how to:

- Monitor a running SSIS package
- Implement Error Handling

Question: In what scenarios would a Try/Catch block in a script help you when a package fails with no error?

Lab: Debugging and Error Handling

- Exercise 1: Debugging a Package
- Exercise 2: Implementing Error Handling
- Exercise 3: Controlling Failure Behavior

Logon information

Virtual machine	NY-SQL-01
User name	Administrator
Password	Pa\$\$w0rd

Estimated time: 90 minutes

Exercise 1: Debugging a Package

Scenario

You are developing an Integration Services package that retrieves employee data from multiple text files and inserts that data into the Employees table in the HumanResources database. You have developed the control flow and data flow, and you now plan to incorporate debugging and error handling functionality into the package.

This exercise's main tasks are:

1. Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment.
2. Open the SSIS_sol6 solution in Business Intelligence Development Studio, and then open the Employees package.
3. Add a breakpoint to the data flow task.
4. Add a breakpoint to the script.

5. Add a data viewer to the package.
 6. Enable Integration Services logging, and display the Log Events pane.
 7. Run the package and monitor events, row counts, data samples, and variable values as they change.
 8. Remove the breakpoints and the data viewer.
- **Task 1: Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment**
- Start 6235A-NY-SQL-01, and log on as **Student** with the password of **Pa\$\$w0rd**.
 - Run the E:\MOD06\Labfiles\Starter\Setup.cmd file.
- **Task 2: Open the SSIS_sol6 solution in Business Intelligence Development Studio**
- The SSIS_sol6.sln file is in the E:\MOD06\Labfiles\Starter\SSIS_sol6 folder.
 - When you have opened the solution, open the **Employees.dtsx** package in SSIS Designer.
- **Task 3: Add a breakpoint to the Data Flow task**
- Add the breakpoint to the **DFT Retrieve employee data** task on the Control Flow design surface.
 - Use the **Break when the container receives the OnPreExecute event** break condition.
- **Task 4: Add a breakpoint to the script**
- Add the breakpoint to the following line of code in the script:

```
Dim sw As StreamWriter
```

► **Task 5: Add a data viewer to the package**

- Add the data viewer to the data flow path that connects the Flat File source to the Data Conversion transformation in the **DFT Retrieve employee data** data flow.
- Use a grid data viewer.

► **Task 6: Enable Integration Services logging, and display the Log Events pane**

- Enable logging on the package object.
- Log **OnError** and **OnPreExecute** events.
- Click **Log Events** on the **SSIS** menu to display the **Log Events** pane.

► **Task 7: Run the package and monitor events, row counts, data samples, and variable values as they change**

- Run the package in debug mode.
- When you reach a break in the control flow, add a watch on the **EmployeeFile** variable, and monitor its value in the **Watch 1** debugging window.
- When you reach a break in the script, monitor the value of the **EmployeeFile** variable in the **Locals** debugging window.
- View row counts in the data flow when you break to view the data viewer window.
- View events in the **Log Events** pane when package execution is complete.

► **Task 8: Remove the breakpoints and the data viewer**

- Delete the data viewer from the data flow path.
- Click **Delete All Breakpoints** on the **Debug** menu to remove all breakpoints.

Results: After this exercise, you should have successfully configured the breakpoints, enabled logging, configured logging, run the package, and verified that the data is viewable at each breakpoint.

Exercise 2: Implementing Error Handling

Scenario

To incorporate error handling, you plan to implement custom logging, catch errors in your script, and control failure behavior to test the error-handling functionality.

This exercise's main tasks are:

1. Create a table in the HumanResources database to log errors.
2. Add an event handler to capture errors generated by the Foreach Loop container and its child tasks.
3. Configure the event handler to prevent propagation.
4. Add an Execute SQL Task to the event handler.
5. Add a Try-Catch block to the script in the Script task.
6. Modify Integration Services logging to log only errors.
7. Configure the employee text files as read-only.
8. Run the package, and view the events in the Log Events pane.
9. View the data in the Employees and LogEvents tables.

► **Task 1: Create a table in the HumanResources database to log errors**

- Run the **LogEventsTable.sql** query file.
- The file is located in the E:\MOD06\Labfiles\Starter\SetupFiles folder.
- You can use the **sqlcmd** utility or SQL Server Management Studio to run the query file.



Note: Type **sqlcmd /?** at a command prompt for help on sqlcmd syntax.

- ▶ **Task 2: Add an event handler to capture errors generated by the Foreach Loop container and its child tasks**
 - Use the **OnError** event.
- ▶ **Task 3: Configure the event handler to prevent propagation**
 - Set the **Propagate** system variable for the event handler to **False**.
- ▶ **Task 4: Add an Execute SQL Task to the event handler**
 - Use the following settings for the Execute SQL task:
 - Name: **SQL Log errors**
 - Description: **Log error data and file name**
 - Connection type: **OLE DB**
 - Connection manager: **HumanResources_cm**
 - SQL source type: **direct input**
 - Use the following INSERT statement to add a row to the **RowCounts** table:

```
INSERT INTO hr.LogEvents
    (ExecGUID, PackageName, SourceName,
     ErrorCode, ErrorDescription, EmployeeFile)
VALUES (?, ?, ?, ?, ?, ?)
```
 - Create six input parameter mappings for the following variables:
 - **ExecutionInstanceGUID** (data type: **NVARCHAR**; parameter name: **0**)
 - **PackageName** (data type: **NVARCHAR**; parameter name: **1**)
 - **SourceName** (data type: **NVARCHAR**; parameter name: **2**)
 - **ErrorCode** (data type: **LONG**; parameter name: **3**)
 - **ErrorDescription** (data type: **NVARCHAR**; parameter name: **4**)
 - **EmployeeFile** (data type: **NVARCHAR**; parameter name: **5**)

► **Task 5: Add a Try-Catch block to the script in the Script task**

- Add a **Try** statement before the first variable declaration.
- Add a **Catch** block after you close and flush the **StreamWriter** object. The **Catch** block should use the **Dts.Events.FireError** method to raise an error if an exception occurs. The error should include a message about not being able to write to the employee file. The **Catch** block should also inform runtime of the failure.
- Add an **End Try** statement after the **Catch** block.

► **Task 6: Modify Integration Services logging to log only errors**

- Log the **OnError** event only. Remove the **OnPreExecute** event.

► **Task 7: Configure the employee text files as read-only**

- The files are in the E:\MOD06\Labfiles\Starter\SetupFiles folder.

► **Task 8: Run the package, and view the events in the Log Events pane**

- Run the package in debug mode.
- View events in the Log Events pane.

► **Task 9: View the data in the Employees and LogEvents tables**

- The tables are in the hr schema of the HumanResources database.
- Use SQL Server Management Studio to view the table contents.
- The **Employees** table should contain 100 rows, and the **LogEvents** table should contain one row.

Results: After this exercise, you should have successfully added a try/catch block, configured the variable to log, run the package, and then confirmed the entries in the hr.LogEvents table.

Exercise 3: Controlling Failure Behavior

Scenario

In this exercise, you will configure the Foreach Loop container and the Script task to control the failure behavior of those tasks and the package. You will run the package each time you modify the settings to view the results of your changes.

This exercise's main tasks are:

1. Configure the **MaximumErrorCount** property of the Foreach Loop container.
 2. Run the package, and view the events in the Log Events pane.
 3. View the data in the Employees and LogEvents tables.
 4. Configure the employee text files as read-write.
 5. Configure the **ForceExecutionResult** property of the Script task.
 6. Run the package, and view the events in the Log Events pane.
 7. View the data in the Employees and LogEvents tables.
- **Task 1: Configure the MaximumErrorCount property of the Foreach Loop container**
- Set the property value to 3.
- **Task 2: Run the package, and view the events in the Log Events pane**
- Run the package in debug mode.
 - View events in the Log Events pane.
- **Task 3: View the data in the Employees and LogEvents tables**
- The **Employees** table should contain 290 rows, and the **LogEvents** table should contain multiple rows, from the previous package execution and from the most recent execution.

- ▶ **Task 4: Configure the employee text files as read-write**
 - The files are in E:\MOD06\Labfiles\Starter\SetupFiles.
 - Highlight the three files, right-click the files, and click **Properties**. In the **Properties** dialog box, clear the **Read-only** check box.
- ▶ **Task 5: Configure the ForceExecutionResult property of the Script task**
 - Set the property value to **Failure**.
- ▶ **Task 6: Run the package, and view the events in the Log Events pane**
 - Run the package in debug mode.
 - View events in the **Log Events** pane.
- ▶ **Task 7: View the data in the Employees and LogEvents tables**
 - The **Employees** table should contain 290 rows, and the **LogEvents** table should not contain any rows related to the forced failure.

Results: After this exercise, you should have successfully modified the package failure behavior, run the package, and verified that the correct rows appear in the database tables.

Lab Shutdown

After you complete the lab, you must shut down the 6235A-NY-SQL-01 virtual machine and discard any changes.

Module Review and Takeaways

- Review Questions
- Best Practices

Review Questions

1. What options should you consider for handling bad data?
2. How would you use a breakpoint and a data viewer to find out why an error is occurring?
3. How do you handle exceptions in a Script task?

Best Practices

Eliminate unneeded logging. Logging is useful for debugging and troubleshooting. However, when deploying completed packages to production, be mindful and careful about the log entries that have been left enabled. Notably, OnPipelineRowsSent is quite verbose, as it logs an entry for every chunk of data sent in each thread that is operational.

If you encounter an Integration Services error number without an accompanying description during package development, you can locate the description in Integration Services Error and Message Reference on MSDN.

Although you cannot directly configure an output as an error output in the Script component for automatic handling of error rows, you can reproduce the functionality of a built-in error output by creating an additional output and using conditional logic in your script to direct rows to this output when appropriate. You may want to imitate the behavior of a built-in error output by adding two additional output columns to receive the error number and the ID of the column in which an error occurred.

Module 7

Implementing Checkpoints and Transactions

Contents:

Lesson 1: Implementing Checkpoints	7-3
Lesson 2: Implementing Transactions	7-10
Lab: Implementing Checkpoints and Transactions	7-17

Module Overview

- Implementing Checkpoints
- Implementing Transactions

Integration Services can restart failed packages from the point of failure. When the failed package is rerun, the checkpoint file is used to restart the package from the point of failure. Packages use transactions to bind the database actions that tasks perform into atomic units, which helps maintain data integrity.

Lesson 1

Implementing Checkpoints

- What Are Checkpoints?
- How Checkpoints Work
- Checkpoint Files
- Enabling Checkpoints

If a package is configured to use checkpoints, information about package execution is written to a checkpoint file. When the failed package is rerun, the checkpoint file is used to restart the package from the point of failure.

What Are Checkpoints?

Checkpoints can help provide the following:

- **Provide restart points within your package**
- **Avoid repeating time-consuming tasks**
- **Avoid repeating downloading and uploading of large files**
- **Avoid repeating loading of large amounts of data**
- **Avoid repeating the aggregation of values**



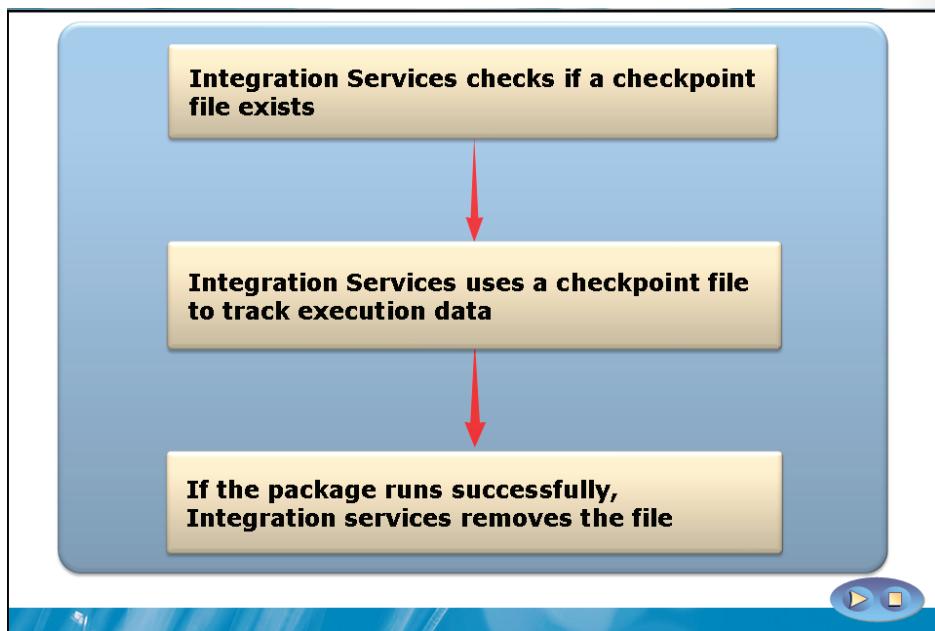
Key Points

Checkpoints have useful features and provide a number of benefits.

- A package can be configured to support restarting the package at checkpoints if one or more of its tasks fail.
- If a package is configured to use checkpoints, information about package execution is written to a checkpoint file.
- When the failed package is rerun, the checkpoint file is used to restart the package from the point of failure.

Question: Have there been times in the past when the use of checkpoints would have helped with deployment problems that you faced.

How Checkpoints Work



Key Points

Integration Services can restart failed packages from the point of failure, instead of rerunning the whole package. If a package is configured to use checkpoints, information about package execution is written to a checkpoint file. When the failed package is rerun, the checkpoint file is used to restart the package from the point of failure.

- If the package runs successfully, the checkpoint file is deleted, and then re-created the next time the package is run.
- The type of container that fails and the implementation of features such as transactions affect the restart point that is recorded in the checkpoint file. The current values of variables are also captured in the checkpoint file.
- If the package fails, Integration Services retains the file. When you restart the package, Integration Services uses the file to determine where to start the execution. If the package runs successfully the second time, Integration Services removes the file; otherwise, it retains the file for another execution attempt.

Checkpoint Files

Checkpoint files are stored in the operating system's file system, separate from Integration Services

Field	Notes
Package ID	<ul style="list-style-type: none">The globally unique identifier (GUID) assigned to the package
Execution results	<ul style="list-style-type: none">The containers and tasks configured as checkpoints and have already run
Variable values	<ul style="list-style-type: none">The current values of the user variables, along with a number of other variables

Configure the appropriate security on the folder where you store the checkpoint file

Key Points

Checkpoint Files have a number of configurable options and provide restart points within your package.

- Checkpoint files are stored as an XML text file.
- Checkpoint files are stored in the operating system's file system, separate from Integration Services.
- The checkpoint file saves the globally unique identifier (GUID) assigned to the package. The ID guarantees that Integration Services is accessing the correct checkpoint file when it reruns the package.
- The checkpoint file tracks the containers and tasks that have been configured as checkpoints and that have already run.

- When you rerun a package that references a checkpoint file, that package begins after the last recorded checkpoint object.
- A Checkpoint file also saves variable values, which may include sensitive data.
- It is important to protect the file location.
- Configure the appropriate security on the folder where you will store the checkpoint file.

Enabling Checkpoints

- 1 Configure the package to use checkpoints by setting the checkpoint properties**

- 2 Configure specific tasks and containers as checkpoints**



Key Points

You configure a package to use checkpoints by setting properties at the package level. You configure the actual checkpoints by setting properties at the container and task level.

- To configure the checkpoints, set the **FailPackageOnFailure** property to **True** for each container or task that you want to configure as a checkpoint.

Demonstration: Configure Checkpoints for Restarting a Failed Package

In this demonstration, you will see how to:

- Configure checkpoints for restarting a failed package

Question: What are the two different values that can be assigned to CheckPointUsage?

Question: What is the difference between the two different CheckPointUsage values?

Lesson 2

Implementing Transactions

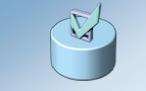
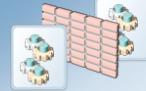
- What Are Transactions?
- Transaction Support in Integration Services
- Transaction Inheritance
- Transaction Scenarios
- Using Native Transactions

Packages use transactions to bind the database actions that tasks perform into atomic units, and by doing this maintain data integrity. Because all the database actions that are part of a transaction are committed or rolled back together, you can ensure that data remains in a consistent state. Transactions in packages can be used to gather the results of several tasks into a single transaction to ensure consistent updates, ensure consistent updates on multiple database servers, guarantee updates in an asynchronous environment, and carry out multiple transactions under the control of a single package.

What Are Transactions?

The slide features a light blue header bar with the title 'What Are Transactions?'. Below it is a white rectangular area divided into four horizontal sections, each containing an icon and text. A decorative blue banner with a play button icon is at the bottom right.

- A transaction is an atomic unit of work**

- A transaction leaves data in a consistent state**

- A transaction is isolated from other concurrent transactions**

- A transaction is durable**


Key Points

A transaction is a logical unit of work made up of one or more tasks.

- For a transaction to complete successfully, every task within that transaction must be successful, or the entire transaction is rolled back.
- A transaction leaves data in a consistent state. Changes must not violate the rules of the structures affected by the transaction.
- A transaction is isolated from other concurrent transactions. The transaction's modifications must be unaffected by other, concurrent transactions.
- This is handled by locking at the data source or destination.
- A transaction is durable. Its effects must be permanent after that transaction is completed. Any modifications you make must persist even if there is system failure.

Transaction Support in Integration Services

Variables with package scope are visible to all containers

Property	Settings	Notes
TransactionOption	<ul style="list-style-type: none">• Required• Supported• NotSupported	<ul style="list-style-type: none">• Specifies whether task supports or implements transaction
IsolationLevel	<ul style="list-style-type: none">• Unspecified• Chaos• ReadUncommitted• ReadCommitted• RepeatableRead• Snapshot• Serializable	<ul style="list-style-type: none">• Specifies isolation level used by task

Transactions are handled by the MSDTC, not by the SSIS engine, unless native transactions are used

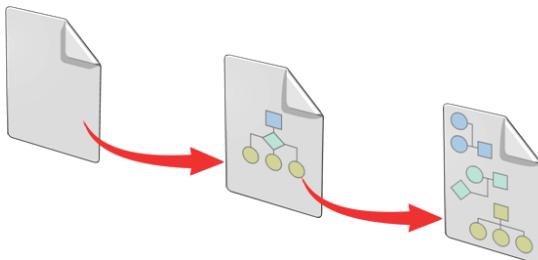
Key Points

You can create transactions in an Integration Services package by setting the transaction-related properties on individual executables, including the package itself.

- Transaction support is handled by the Distributed Transaction Coordinator, a Windows service.
- Each executable supports two transaction-related properties: **TransactionOption** and **IsolationLevel**.
- Each container has its own Variables collection. When a new variable is created, it is within the scope of its parent container.
- Because the package container is at the top of the container hierarchy, variables with package scope function like global variables, and are visible to all containers within the package.

Transaction Inheritance

Transaction Option	Transaction Behavior
Required	<ul style="list-style-type: none">Start a new transaction
Supported	<ul style="list-style-type: none">Join the parent transaction
Not Supported	<ul style="list-style-type: none">Do not join or start a transaction



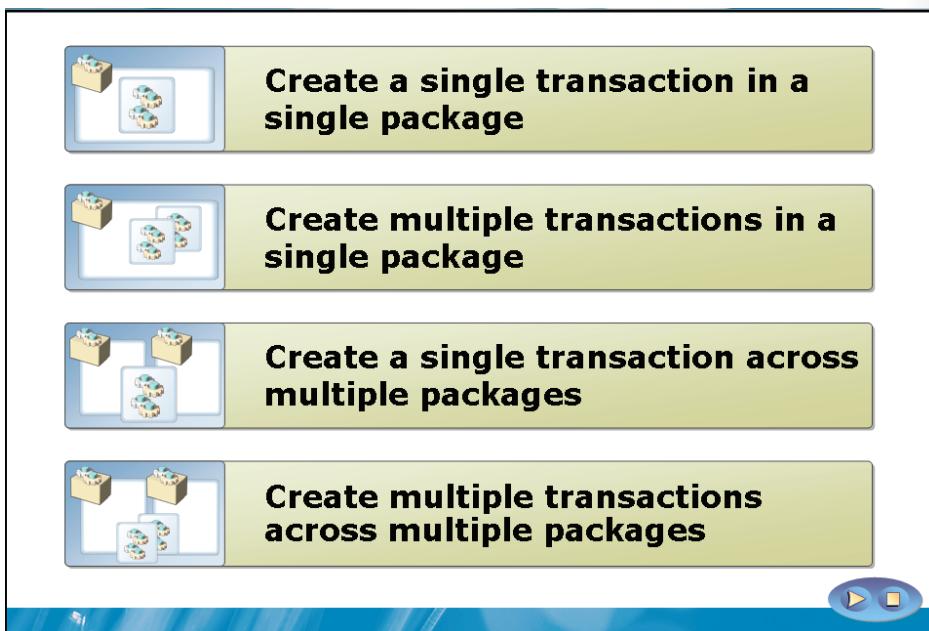
Key Points

You can configure a container or task to start a new transaction, join the parent's transaction, or not participate in a transaction.

You configure this behavior by setting the **TransactionOption** property on the specific task or container:

- The Required TransactionOption configures a task or container to start a new transaction.
- The Supported TransactionOption configures a task or container to join the parent's transaction.
- The NotSupported TransactionOption configures a task or container to not participate in a transaction.

Transaction Scenarios



Key Points

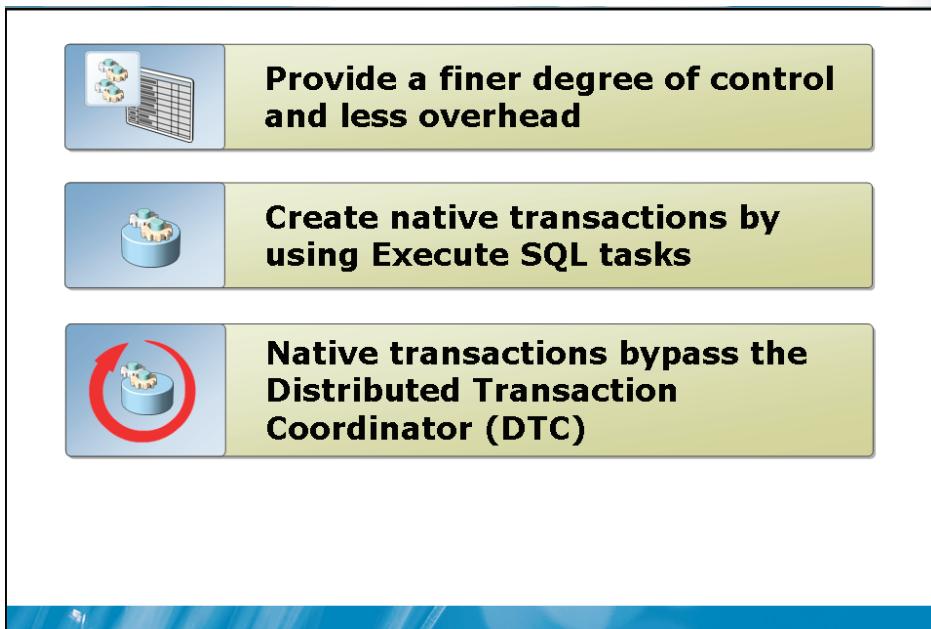
You use the **TransactionOption** property to control which executables start transactions and which ones participate in them.

- To create a single transaction in a single package, set the **TransactionOption** property to **Required** for the top-level executable, and set the property to **Supported** for each child component within that executable.
- To create multiple transactions in a single package, set the **TransactionOption** property to **Required** for all executables that should start a transaction, and set the property to **Supported** for each child component within a transacted container.

- In multiple-package configurations, you use a parent package to run other packages. To create a single transaction across multiple packages, set the **TransactionOption** property to **Required** for the executable in the parent package that should start the transaction, and set the property to **Supported** for each child component within that executable, including any Execute Package tasks.
- To create multiple transactions across multiple packages, set the **TransactionOption** property to **Required** for all executables in the parent package that should start a transaction, and set the property to **Supported** for each child component within a transacted container, including any Execute Package tasks.

Question: Which transaction scenario would be most useful to your organization?

Using Native Transactions



Key Points

Native transactions implement a transaction on a relational data source, rather than within Integration Services.

- You configure native transactions by encapsulating your workflow in a set of Execute SQL tasks that begin and commit the transaction.
- These transactions bypass the DTC.
- Native transactions can provide a finer degree of control over an Integration Services transaction and generally require less overhead.
- By using a native transaction, you can take advantage of all the transaction capabilities within the relational database used as the data source.

Lab: Implementing Checkpoints and Transactions

- Exercise 1: Implementing Checkpoints in a Package
- Exercise 2: Implementing Transactions in a Package
- Exercise 3: Implementing a Native Transaction

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$wOrd

Estimated time: 60 minutes

Exercise 1: Implementing Checkpoints in a Package

Scenario

Adventure Works is planning a rollout of new database solution for the Human Resources department. It is important that this rollout go smoothly and to completion, so it has been decided to use checkpoints to make sure that the package is fully deployed.

You are developing an Integration Services solution that retrieves employee data from text files and inserts that data into the HumanResources database. You have already developed the control flow and data flow. You now want to implement checkpoints and transactions into the packages.

The senior database developer has asked you to perform the following task:

- Implement checkpoints into the packages.

In this exercise, you must implement checkpoints into the packages.

The main tasks for this exercise are as follows:

1. Configure the lab environment.
2. View the database data.
3. Open an SSIS solution.
4. Configure the checkpoint properties on the package.
5. Configure the **FailPackageonFailure** property on the tasks in the Sequence container.
6. Run the package.
7. View the database data.
8. View the output file.
9. Configure the output file as read-write.
10. Re-run the package.
11. View database data and the output file.
12. Verify that an email message was sent.

► **Task 1: Configure the lab environment**

- Start 6235A-NY-SQL-01 and logon as **Student** using the password **Pa\$\$w0rd**.
- Start **the Simple Mail Transport Protocol (SMTP)** service.
- Start the Server Management Studio and connect to the server.
- Run the setup code.
 - Name: **Setup.cmd**
 - Location: **E:\Mod07\Labfiles\Solution**

► **Task 2: View the database data**

- Start the Server Management Studio and connect to the server.
- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees1**
 - Expected results: **empty**

► **Task 3: Open an SSIS solution**

- Start SQL Server Business Intelligence Development Studio.
- Open and examine package solution.
 - Folder: E:\Mod07\Labfiles\Starter\SSIS_sol7
 - Solution: **SSIS_sol7.sln**
 - Package: **Employees1.dtsx**

► **Task 4: Configure the checkpoint properties on the package**

- Configure the checkpoint properties on the package.
 - CheckpointFileName: E:\Mod07\Labfiles\Starter\Employees\EmployeeCheckpoint.txt
 - CheckpointUsage list: **IfExists**
 - SaveCheckpoints list: **True**

- ▶ **Task 5: Configure the FailPackageonFailure property on the tasks in the Sequence container**
 - Configure the SQL Truncate Employees1 task in the Sequence container.
 - FailPackageOnFailure: **True**
 - Configure the DFT Retrieve employee data task in the Sequence container.
 - FailPackageOnFailure: **True**
 - Configure the SCR Add timestamp task in the Sequence container.
 - FailPackageOnFailure: **True**
- ▶ **Task 6: Run the package**
 - Start debugging and verify results.
 - Verify SQL Truncate Employees1 Execute SQL task: **green**
 - Verify DFT Retrieve employee data Data Flow task: **green**
 - Verify SEQ Process employee data Sequence container task: **red**
 - Verify SCR Add timestamp Script task: **red**
 - Stop debugging.
- ▶ **Task 7: View the database data**
 - Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees1**
 - Expected results: **290 rows**
- ▶ **Task 8: View the output file**
 - Verify that the file contains checkpoint data.
 - Folder: **E:\Mod07\Labfiles\Starter\Employees**
 - File: **EmployeeCheckpoint.txt**

► **Task 9: Configure the output file as read-write**

- Configure the E:\Mod07\Labfiles\Starter\Employees\Employees1.txt file as read-write.

► **Task 10: Re-run the package**

- Start debugging and verify results.
 - Verify SQL Truncate Employees1 Execute SQL task: **does not change color**
 - Verify DFT Retrieve employee data Data Flow task: **does not change color**
 - Verify SEQ Process employee data Sequence container task: **green**
 - Verify SCR Add timestamp Script task: **green**
- Stop debugging.

► **Task 11: View database data and the output file**

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees1**
 - Expected results: **290 rows**
- Verify that a timestamp has been added to the employee file.
 - Folder: **E:\Mod07\Labfiles\Starter\Employees**
 - File: **Employees1.txt**

► **Task 12: Verify that an email message was sent**

- Verify that the SMT Send e-mail on success Send Mail task has sent an e-mail message.
 - Mail folder: **C:\inetpub\mailroot\Drop**
 - Alternate mail folder: **C:\inetpub\mailroot\Queue**

Results: After this exercise, the checkpoints have been implemented in the package.

Exercise 2: Implementing Transactions in a Package

Scenario

In order to increase manageability and reliability of the package deployment process, the IT Director has decided to use transactions in package deployment. After testing the package, the senior database developer has asked you to perform the following task:

- Implement transactions in the package.

In this exercise, you need to implement transactions in the package.

The main tasks for this exercise are as follows:

1. View the database data.
2. Configure the Sequence container to start a transaction.
3. Configure the **ValidateExternalMetadata** property on the OLE DB destination.
4. Run the package.
5. View the database data.
6. Configure the output file as read-write.
7. Re-run the package.
8. View the database data and the output file.
9. Verify that an email message was sent.

► Task 1: View the database data

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees2**
 - Expected results: **10 rows**

► **Task 2: Configure the Sequence container to start a transaction**

- Examine the Employees2.dtsx package.
- Configure the container properties.
 - Container: **SEQ Process employee data**
 - Property: **TransactionOption**
 - Value: **Required**

► **Task 3: Configure the ValidateExternalMetadata property on the OLE DB destination**

- Configure the DFT Retrieve employee data task.
- Task: **OLE Load employee data**
- Property: **ValidateExternalMetadata**
- Value: **False**

► **Task 4: Run the package**

- Switch to the **Control Flow** tab for the Employees2.dtsx package.
- Start debugging and verify results.
 - Verify SQL Truncate Employees2 Execute SQL task: **green**
 - Verify DFT Retrieve employee data Data Flow task: **green**
 - Verify SEQ Process employee data Sequence container task: **red**
 - Verify SCR Add timestamp Script task: **red**
- Stop debugging.

► **Task 5: View the database data**

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees2**
 - Expected results: **10 rows**

► **Task 6: Configure the output file as read-write**

- Configure the E:\Mod07\Labfiles\Starter\Employees\Employees2.txt file as read-write.

► **Task 7: Re-run the package**

- Start debugging and verify results.
 - Verify SEQ Process employee data Sequence container task: **green**
 - Verify SQL Truncate Employees2 Execute SQL task: **green**
 - Verify DFT Retrieve employee data Data Flow task: **green**
 - Verify SCR Add timestamp Script task: **green**
 - Verify SEM Send e-mail on success Send Mail task: **green**
- Stop debugging.

► **Task 8: View the database data and the output file**

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees2**
 - Expected results: **290 rows**
- Verify that a timestamp has been added to the employee file.
 - Folder: **E:\Mod07\Labfiles\Starter\Employees**
 - File: **Employees2.txt**

► **Task 9: Verify that an email message was sent**

- Verify that the SMT Send e-mail on success Send Mail task has sent an e-mail message.
 - Mail folder: C:\inetpub\mailroot\Drop
 - Alternate mail folder: C:\inetpub\mailroot\Queue

Results: After this exercise, the transactions have been implemented in the package.

Exercise 3: Implementing a Native Transaction

Scenario

Additional research has found that native transactions can provide a finer degree of control over an Integration Services transaction and generally require less overhead. After testing the package, the senior database developer has asked you to perform the following task:

- Implement a native transaction.

In this exercise, you need to implement a native transaction.

The main tasks for this exercise are as follows:

1. View the database data.
2. Add an Execute SQL Task to the Sequence container in the package.
3. Add another Execute SQL Task to the Sequence container.
4. Configure the **RetainSameConnection** property on the connection manager.
5. Run the package.
6. View the database data.
7. Configure the output file as read-write.
8. Re-run the package.
9. View database data and the output file.
10. Verify that an email message was sent.

► Task 1: View the database data

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees3**
 - Expected results: **10 rows**

► **Task 2: Add an Execute SQL Task to the Sequence container in the package**

- Examine the Employees2.dtsx package.
- Add an **Execute SQL Task** from the Toolbox to inside the SEQ Process employee data Sequence container above the existing tasks.
- Connect the task to the SQL Truncate Employees3 Execute SQL task.
- Configure the new Execute SQL Task name.
 - Page: **General**
 - Property: **Name**
 - Value: **SQL Begin transaction**
- Configure the new Execute SQL Task description.
 - Page: **General**
 - Property: **Description**
 - Value: **Start transaction at beginning of container**
- Configure the new Execute SQL Task connection.
 - Page: **General**
 - Property: **Connection list**
 - Value: **HumanResources_cm**
- Configure the new Execute SQL Task SQL statement.
 - Page: **General**
 - Property: **SQLStatement**

```
BEGIN TRANSACTION EmpTran  
GO
```

► **Task 3: Add another Execute SQL Task to the Sequence container**

- Add an **Execute SQL Task** from the Toolbox to inside the SEQ Process employee data Sequence container below the existing tasks.
- Connect the task to the SCR Add timestamp Script task.
- Configure the new Execute SQL Task name.
 - Page: **General**
 - Property: **Name**
 - Value: **SQL Commit transaction**
- Configure the new Execute SQL Task description.
 - Page: **General**
 - Property: **Description**
 - Value: **Commit transaction at end of container**
- Configure the new Execute SQL Task connection.
 - Page: **General**
 - Property: **Connection list**
 - Value: **HumanResources_cm**
- Configure the new Execute SQL Task SQL statement.
 - Page: **General**
 - Property: **SQLStatement**

```
COMMIT TRANSACTION EmpTran  
GO
```

► **Task 4: Configure the RetainSameConnection property on the connection manager**

- Configure the HumanResources_cm connection.
 - Connection: **HumanResources_cm**
 - Property: **RetainSameConnection**
 - Value: **True**

► **Task 5: Run the package**

- Start debugging and verify results.
 - Verify SEQ Process employee data Sequence container task: **red**
 - Verify SQL Begin Transaction Execute SQL task: **green**
 - Verify SQL Truncate Employees3 Execute SQL task: **green**
 - Verify DFT Retrieve employee data Data Flow task: **green**
 - Verify SCR Add timestamp Script task: **red**
- Stop debugging.

► **Task 6: View the database data**

- Examine the contents of the employees table.
 - Database: **HumanResources**
 - Table: **hr.Employees3**
 - Expected results: **10 rows**

► **Task 7: Configure the output file as read-write**

- Configure the E:\Mod07\Labfiles\Starter\Employees\Employees3.txt file as read-write.

► **Task 8: Re-run the package**

- Start debugging and verify results.
 - Verify SEQ Process employee data Sequence container task: **green**
 - Verify SQL Begin Transaction Execute SQL task: **green**
 - Verify SQL Truncate Employees3 Execute SQL task: **green**
 - Verify DFT Retrieve employee data Data Flow task: **green**
 - Verify SCR Add timestamp Script task: **green**
 - Verify SEM Send e-mail on success Send Mail task: **green**
- Stop debugging.

► **Task 9: View database data and the output file**

- Examine the contents of the employees table
 - Database: **HumanResources**
 - Table: **hr.Employees3**
 - Expected results: **290 rows**
- Verify that a timestamp has been added to the employee file.
 - Folder: **E:\Mod07\Labfiles\Starter\Employees**
 - File: **Employees3.txt**

► **Task 10: Verify that an email message was sent**

- Verify that the SMT Send e-mail on success Send Mail task has sent an e-mail message.
 - Mail folder: **C:\inetpub\mailroot\Drop**
 - Alternate mail folder: **C:\inetpub\mailroot\Queue**
- Turn off 6235A-NY-SQL-01 and delete changes.

Results: After this exercise, the native transactions have been implemented in the package.

Module Review and Takeaways

- Review Questions
- Best Practices

Review Questions

1. Where are checkpoint files stored?
2. What is a transaction?
3. What are the tree settings available for the **TransactionOption** property?
4. What are native transactions?

Best Practices for Implementing Checkpoints and Transactions

Supplement or modify the following best practices for your own work situations:

- Avoid using checkpoints and transactions in the same package. Using checkpoints and transactions in the same package could cause unexpected results. For example, when a package fails and restarts from a checkpoint, the package might repeat a transaction that has already been successfully committed.
- The SHUTDOWN WITH NOWAIT statement shuts down SQL Server without executing a checkpoint in each database. This may cause the subsequent restart to take a longer time than usual to recover the databases on the server.
- In general, the amount time required for a checkpoint operation increases with the number of dirty pages that the operation must write. To minimize the performance impact on other applications, SQL Server by default adjusts the frequency of writes that a checkpoint operation performs. SQL Server uses this strategy for automatic checkpoints and for any CHECKPOINT statement that does not specify a checkpoint_duration value. Decreasing the write frequency increases the time the checkpoint operation requires to complete.
- You can use checkpoint_duration to request that the checkpoint operation complete within a specific amount of time. The performance impact of using checkpoint_duration depends on the number of dirty pages, the activity on the system, and the actual duration specified. In general, a short checkpoint_duration will increase the resources devoted to the checkpoint, while a long checkpoint_duration will reduce the resources devoted to the checkpoint. SQL Server always completes a checkpoint if possible, and the CHECKPOINT statement returns immediately when a checkpoint completes. Therefore, in some cases, a checkpoint may complete sooner than the specified duration or may run longer than the specified duration.

Module 8

Configuring and Deploying Packages

Contents:

Lesson 1: Package Configurations	8-3
Lesson 2: Preparing and Deploying Packages	8-15
Lab: Deploying Packages	8-30

Module Overview

- Package Configurations
- Preparing and Deploying Packages

Microsoft® SQL Server® Integration Services provides tools that make it easy to deploy packages to another computer. The deployment tools also manage any dependencies, such as configurations and files that the package needs. In this module, you will learn how to use these tools to install packages and their dependencies on a target computer.

Lesson 1

Package Configurations

- What Are Package Configurations?
- What Are the Package Configuration Formats?
- What Are the Types of Package Configurations?
- What Are Property Expressions?
- Preparing for Package Configuration
- Creating a Package Configuration

SQL Server Integration Services provides package configurations that you can use to update the values of properties at run time. Typically, you create package properties on the package objects during package development, and then add the configuration to the package. When the package runs, it gets the new values of the property from the configuration. Configurations make it easier to move packages from a development environment to a production environment. Configurations are useful when you deploy packages to many different servers. Configurations make packages more flexible.

What Are Package Configurations?

Collection or set of property settings that can be configured at deployment or runtime

Helps move packages from a development to a production environment

Helps you deploy packages to different servers

Supported in multiple formats

Key Points

A package configuration is a property/value pair that is added to a completed package.

- Package configurations enable a package to dynamically access property values at run time.
- A package can retrieve property settings from a number of different source types, such as an XML file or Microsoft SQL Server database.
- You can update settings on any executable (including the package object), connection manager, log provider, or variable within a package.
- When Integration Services runs the package, it extracts the property value from a source that is based on the configuration type.

- You should use package configurations when you are not sure of all the operational parameters that will be needed to run the package.
- Configurations make it easier to move packages from a development environment to a production environment.

Question: Have you used and/or configured packages in your current organization?

What Are the Package Configuration Formats?

Format	Notes
XML	<ul style="list-style-type: none">You can store one or more configuration settings to an XML file
Environment variable	<ul style="list-style-type: none">You can store a single configuration setting in an environmental variable
Registry entry	<ul style="list-style-type: none">You can store a single configuration setting in a registry entry
Parent package variable	<ul style="list-style-type: none">You can store a single configuration setting in a variable at the package object scope
SQL Server	<ul style="list-style-type: none">You can store one or more configuration settings in a table in a SQL Server database

Key Points

Integration Services supports several different methods of storing package configurations. Each configuration is a property/value pair. The configurations are included when you create a package deployment utility for installing packages.

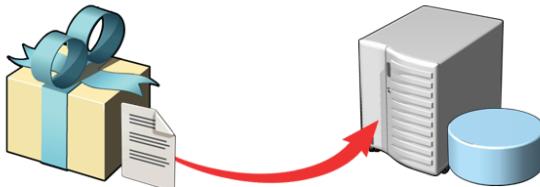
- You can store one or more configuration settings to an XML file.
- You can store settings in an existing XML file or create a new file. If you save settings to an existing XML file, you can add new settings or overwrite old ones.
- You can store a single configuration setting in an environmental variable.
- You can store a single configuration setting in a registry entry. You can use either an existing key or can create a new key.
- You can store a single configuration setting in a user variable created at the scope of the package object.

- You can store one or more configuration settings in a table in a SQL Server database.
- When you create the configuration settings, you must specify a configuration filter. This is a name that you provide for the configuration. It maps to the ConfigurationFilter column in the configuration table.
- The XML configuration file and SQL Server configuration types can include multiple configurations.

Question: What formats of package configurations have you found to be most useful?

What Are the Types of Package Configurations?

Type	Definition	Best Practices
Direct	Creates a direct link between the configuration and the object property	A better choice when the location of the source does not change
Indirect	Uses environment variables	A better choice when the location of the configuration can change for each deployment of a package



Key Points

Integration Services provides both direct and indirect package configurations.

- If you specify configurations directly, Integration Services creates a direct link between the configuration item and the package object property.
- Direct configurations are a better choice when the location of the source does not change.
- Indirect configurations use environment variables. Instead of specifying the configuration setting directly, the configuration points to an environment variable, which in turn, contains the configuration value.

- Using indirect configurations is a better choice when the location of the configuration can change for each deployment of the package.
- Indirect configuration locations are useful when the package will be deployed into a production environment where the configuration location is unknown or subject to change.

Question: Have you used direct or indirect or both types of package configurations?

What Are Property Expressions?

What are Property Expressions?

- Define property values
- Can include variables

Property expressions can be updated in different ways:

- User-defined variables can be included in configurations and then updated when the package is deployed
- System variables in expressions are updated at run time
- Date and time functions are evaluated at run time
- Variables in expressions can be updated by scripts



Key Points

Property expressions are expressions that are assigned to a property, to enable dynamic update of the property at run time.

- You can configure property expressions on any property associated with a package executable, event handler, connection manager, or log provider.
- Property expressions define property values. When you configure a component's properties, you can associate an expression with any of those properties.
- Property expressions can include variables. You can provide a value to the variable through the package, or you can retrieve the value through a package configuration at run time.

- User-defined variables can be included in package configurations and then updated when the package is deployed. At run time, the property expression is evaluated using the updated variable value.
- System variables that are included in expressions are updated at run time, which changes the results of the property evaluation.
- Date and time functions are evaluated at run time and provide the updated values to property expressions.
- Variables in expressions can be updated by the scripts that the Script task and the Script component run.

Question: Have you used variables and/or functions when configuring and deploying packages?

Preparing for Package Configuration

There are three steps to prepare package configurations:

1 Identify properties modified or exposed by configuration

2 Identify properties to be configured

3 Define variables to be referenced in configuration

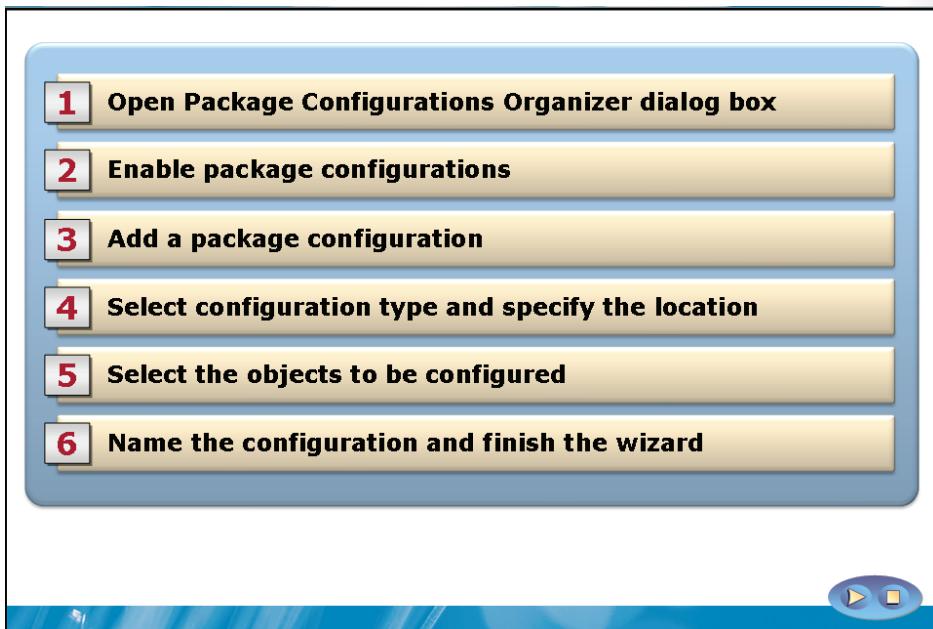


Key Points

There are three steps to prepare package configurations:

- Package configurations are created using the Package Configuration Organizer dialog box and the Package Configuration Wizard.

Creating a Package Configuration



Key Points

Package configurations are created using the Package Configuration Organizer dialog box and the Package Configuration Wizard.

- The Package Configuration Organizer dialog box can be used to: enable packages to use configurations, add and delete configurations, and set the preferred order in which configurations should be loaded.
- From the Package Configuration Organizer dialog box, you run the Package Configuration Wizard, which guides you through the steps to create a configuration.
- On the wizard pages, choose the configuration type, select whether you want to access the configuration directly or use environment variables, and select the properties to save in the configuration.

Demonstration: Creating a Package Configuration

In this demonstration, you will see how to create a package configuration by:

- Adding a variable to a package
- Configuring a property expression for a task
- Adding a package configuration
- Viewing the package configuration file

Question: In this demonstration, what property value did we use, and what was it used for?

Lesson 2

Preparing and Deploying Packages

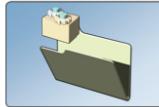
- Package Deployment Options
- Overview of the Package Deployment Process
- Preparing a Package for Deployment
- What Are the Components of the Deployment Utility?
- Creating a Deployment Utility
- Options for Installing a Package
- Installing a Package Using the Installation Wizard
- Troubleshooting Package Deployment

You need to perform several tasks to prepare for package deployment. You will create a new Integration Services project in Business Intelligence Development Studio and add existing packages and data files to the project. You will learn about package dependencies such as log files and about other interesting features of the packages.

Package Deployment Options



To an instance of SQL Server



To the file system



To the SSIS package store



Key Points

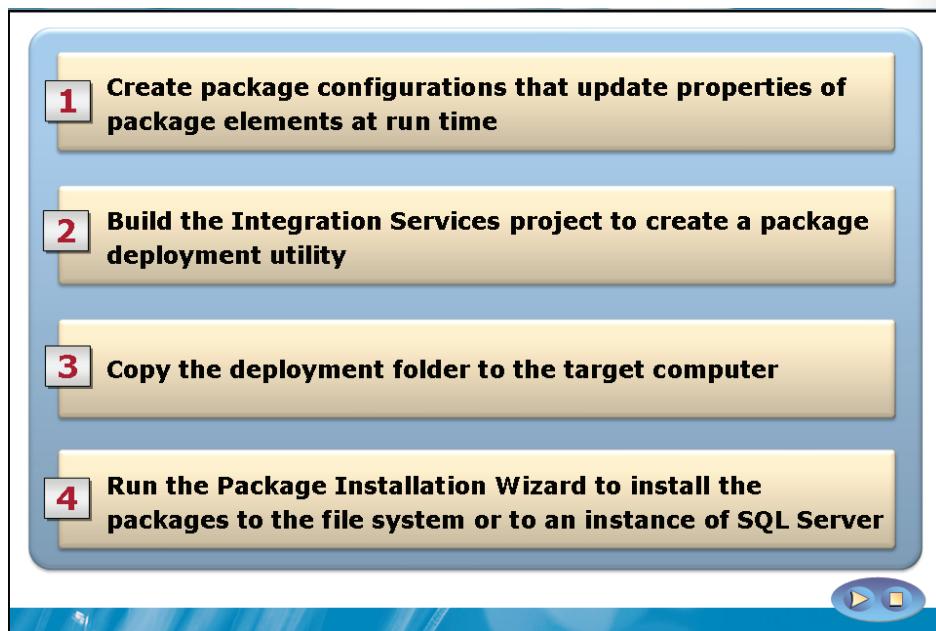
Packages can be deployed to an instance of SQL Server, to the file system, or the SSIS Package.

- When you deploy packages to an instance of Integrations Services, you must decide whether you will store the packages in SQL Server or in the file system.
- You can deploy your packages to SQL Server. SQL Server stores Integration Services packages in the msdb system database, and stores package dependencies in a file folder. When you deploy a package to SQL Server, you must specify the instance of SQL Server and the type of authentication.

- You can deploy your packages to the file system with the SSIS Package (*.dtsx file). Integration Services then accesses the package files and their dependencies from the folder where you deployed the packages.
- You can deploy your packages to the SSIS package store. The SSIS package stores are directories related to the SSIS installation.

Question: What deployment options have you used in the past?

Overview of the Package Deployment Process



Key Points

SQL Server Integration Services includes tools and wizards that make it simple to deploy packages from the development computer to the production server or to other computers.

- The first step is optional and involves creating package configurations that update properties of package elements at run time. The configurations are automatically included when you deploy the packages.
- The second step is to build the Integration Services project to create a package deployment utility. The deployment utility for the project contains the packages that you want to deploy.
- The third step is to copy the deployment folder, which was created when you built the Integration Services project, to the target computer.
- The fourth step is to run the Package Installation Wizard on the target computer to install the packages to the file system or to a SQL Server instance.

Preparing a Package for Deployment

To prepare a package for deployment:

Configure package execution settings

Optimize the package for production

Remove unnecessary elements



Key Points

Before you deploy any packages to an instance of Integration Services, you should prepare those packages by configuring the execution settings, optimizing the packages for a production environment, and removing any unnecessary elements.

- Some components have properties that you can use to optimize their performance. In particular, Data Flow tasks should usually have their **RunInOptimized** mode property set to **True** so that unused columns, outputs, and components are removed from the execution plan. Additionally, you can change the **DefaultBufferSize**, **DefaultBufferMaxRows**, and **MaxThreads** properties of a Data Flow task to optimize its performance.

Demonstration: Prepare a Package for Deployment

In this demonstration, you will see how to prepare a package for deployment by:

- Setting Package Properties and Removing Unnecessary Objects

Question: What does PackagePriorityClass do?

What Are the Components of the Deployment Utility?



A folder that contains files needed to deploy packages in an IS project on a different server



Configure project properties to enable deployment utility



The manifest file is an XML file that lists the packages, the configurations, and other files in the project



Key Points

SQL Server Integration Services packages can be deployed using a deployment utility, using the import and export package features in SQL Server Management Studio, or by saving a copy of the package in Business Intelligence Development Studio.

- Only the deployment utility can deploy multiple packages. The deployment utility also automatically includes the package dependencies (for example, configurations) and the files that contain supporting information (for example, documentation).
- An Integration Services package deployment utility consists of a folder that contains a copy of every package in an SSIS project, copies of package dependencies (such as a configuration file), and a deployment manifest file.

- A deployment utility is a program for Integration Services packages. Integration Services creates the utility automatically when you build the package; however, you must set the project's **CreateDeploymentUtility** property to **True** before Integration Services will create the utility.
- When you build an Integration Services project, a manifest file, <project name>.SSISDeploymentManifest.xml, is created and added, together with copies of the project packages and package dependencies, to the bin\Deployment folder in the project, or to the location specified in the **DeploymentOutputPath** property. The manifest file lists the packages, the package configurations, and any miscellaneous files in the project.

Creating a Deployment Utility

To create a package deployment utility:

- 1. Open the Integration Services project solution**
- 2. Right-click the project and click Properties**
- 3. In the Property Pages dialog box, click Deployment Utility**
- 4. Set AllowConfigurationChanges to True**
- 5. Set CreateDeploymentUtility to True**
- 6. Optionally, update the location of the deployment utility**
- 7. Click OK**
- 8. In Solution Explorer, build the project**
- 9. View the build progress and errors in the Output window**

Key Points

To create a deployment utility, you must configure the project's deployment properties.

- You configure the properties in the Property Pages dialog box. To access the dialog box, right-click the project name in Solution Explorer, and then click Properties.
- The **AllowConfigurationChanges** property determines whether or not package configurations can be changed during deployment.
- The **CreateDeploymentUtility** property determines whether or not the build process should generate a deployment utility. You must set this property to **True** before building the project in order to create a deployment utility.
- The **DeploymentOutputPath** property determines the file system location where the deployment utility is to be created.

Demonstration: Create a Package Deployment Utility

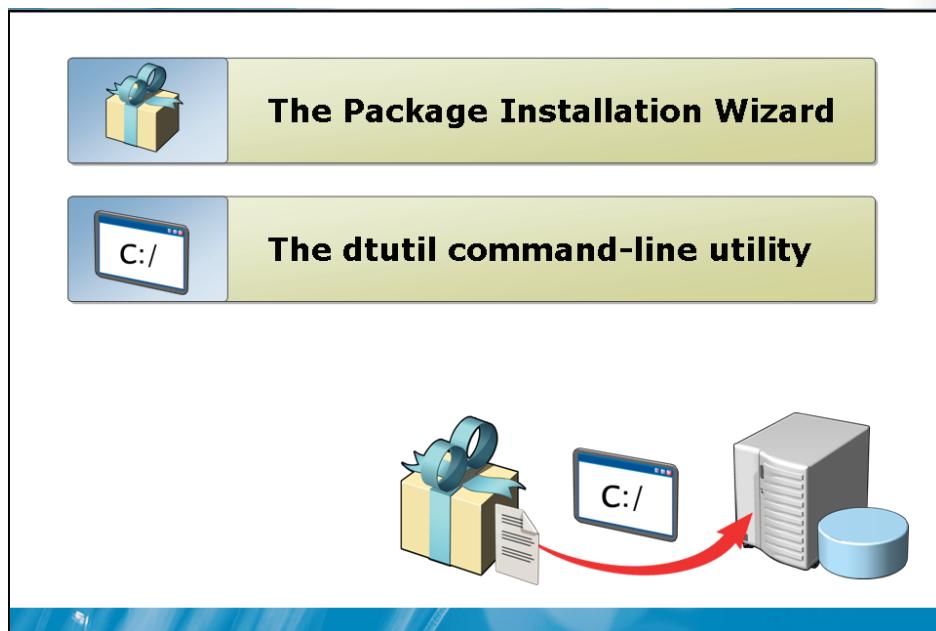
In this demonstration, you will see how to create a package deployment utility by:

- Configuring the Deployment Properties
- Building the Package
- Viewing the Deployment Utility Manifest

Question: What does the `AllowConfigurationChanges` property do?

Question: What does the `CreateDeploymentUtility` property do?

Options for Installing a Package



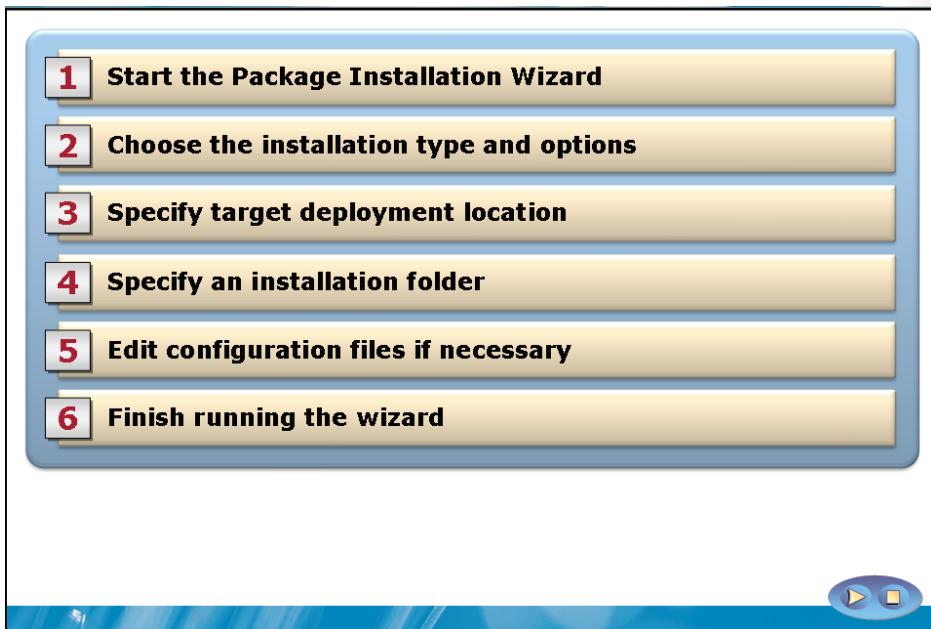
Key Points

SQL Server Integration Services provides tools that make it easy to deploy packages.

- The deployment tools manage any dependencies, such as configurations and files that the package requires.
- The **Package Installation Wizard** takes you through the process of deploying your packages. You launch the wizard by using the deployment manifest file, which contains the information and references you need to complete the wizard.
- The **dtutil** command-line utility can also be used to deploy packages. The dtutil command prompt utility is used to manage SQL Server Integration Services packages. The utility can copy, move, delete, or verify the existence of a package.

Question: Which tool have you used for deploying packages?

Installing a Package Using the Installation Wizard



Key Points

SQL Server Integration Services provides tools that make it easy to deploy packages.

- The deployment tools manage any dependencies, such as configurations and files that the package requires.
- The Package Installation Wizard takes you through the process of deploying your packages. You launch the wizard by using the deployment manifest file, which contains the information and references you need to complete the wizard.

Demonstration: Deploy Packages to the File System Using the Package Installation Wizard

In this demonstration, you will see how to:

- Deploy packages to the file system using the package installation wizard

Question: What are the advantages of using the Package Installation Wizard to deploy the package?

Troubleshooting Package Deployment

Techniques for troubleshooting deployed packages:

Catch and handle package errors by using event handlers

Track the steps of package execution by using logging

To test the deployed packages:

Verify the results of the DataTransfer package

Verify the results of the LoadXMLData package

Techniques to avoid problems with running packages:

Ensure data integrity by using transactions

Restart packages from the failure point using checkpoints

Key Points

Integration Services includes features and tools that you can use to troubleshoot packages when you execute them after they have been completed and deployed.

- At design time, Business Intelligence Development Studio provides breakpoints to pause package execution, the Progress window, and data viewers to watch your data as it passes through the data flow.
- You can respond to the many events that are raised by the package and the objects in the package by using event handlers.
- You can use the error output available on many data flow components to direct rows that contain errors to a separate destination for later analysis.
- Send rows that contain errors to a separate destination such as an error table or a text file. The error output automatically adds two numeric columns that contain the number of the error that caused the row to be rejected, and the ID of the column in which the error occurred.

- You can make it easier to analyze the error output by adding descriptive information in addition to the two numeric identifiers that are supplied by the error output.
- You can track much of what occurs in your running packages by enabling logging. Log providers capture information about the specified events for later analysis, and save that information in a database table, a flat file, an XML file, or another supported output format.
- You can refine the logging output by selecting only the events and only the items of information that you want to capture. There are logging messages that help you troubleshoot a package's interaction with external data sources.

Lab: Deploying Packages

- Exercise 1: Creating a Package Configuration
- Exercise 2: Preparing a Package for Deployment
- Exercise 3: Deploying a Package

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$w0rd

Estimated time: 60 minutes

Exercise 1: Creating a Package Configuration

Scenario

You have developed an Integration Services package that retrieves employee data from a text file and inserts that data into the HumanResources database. You now plan to create a package configuration, prepare the package for deployment, and deploy the package.

In this exercise, you must create a package configuration.

The main tasks for this exercise are as follows:

1. Configure the lab environment.
2. Open an SSIS solution.
3. Add two variables to the package.
4. Configure property expressions for a task.

5. Add a package configuration.
 6. View the package configuration file.
- **Task 1: Configure the lab environment**
- Start 6235A-NY-SQL-01 and logon as **Student** using the password **Pa\$\$w0rd**.
 - Run the setup code.
 - Name: **Setup.cmd**
 - Location: E:\Mod08\Labfiles\Starter
- **Task 2: Open an SSIS solution**
- Start SQL Server Business Intelligence Development Studio.
 - Close the default Start page.
 - Open and examine package solution.
 - Folder: E:\MOD08\Labfiles\Starter\SSIS_sol8
 - Solution: **SSIS_sol8.sln**
 - Package: **Employees.dtsx**
- **Task 3: Add two variables to the package**
- Add the first variables to the package.
 - Name: **BCCLine**
 - Scope: **Employees**
 - Data Type: **String**
 - Value: **sqlserver@adventure-works.com**
 - Add the second variables to the package.
 - Name: **ToLine**
 - Scope: **Employees**
 - Data Type: **String**
 - Value: **Administrator @adventure-works.com**

► **Task 4: Configure property expressions for a task**

- Configure the first SEM Send e-mail on success Send Mail task property expression.
 - Property: **BCCLine**
 - Expression: **@[User::BCCLine]**
- Configure the second SEM Send e-mail on success Send Mail task property expression.
 - Field: **Expressions**
 - Property: **ToLine**
 - Expression: **@[User::ToLine]**

► **Task 5: Add a package configuration**

- Start the package configuration wizard.
- Set the Configuration Type page options.
 - Configuration type: **XML configuration file**
 - Configuration file name: **E:\MOD08\Labfiles\Starter\Employees\SendMail.dtsConfig**
- Set the Select Properties to Export page options.
 - Checkbox: **Variables | BCCLine | Properties | Value**
 - Checkbox: **Variables | ToLine | Properties | Value**
- Set the Completing the Wizard page options.
 - Configuration name: **SendMail**
- Review the settings, and then finish the wizard.

► **Task 6: View the package configuration file**

- Verify the XML data file that Integration Services created.
 - Folder: E:\MOD08\Labfiles\Starter\Employees
 - File: SendMail.dtsConfig

Results: After this exercise, you should have created a package configuration.

Exercise 2: Preparing a Package for Deployment

The main tasks for this exercise are as follows:

1. Set package properties and remove unnecessary objects.
2. Configure the deployment properties and build the package.
3. View the deployment utility manifest.

► **Task 1: Set package properties and remove unnecessary objects**

- Set the control flow design surface properties.
 - **PackagePriorityClass: AboveNormal**
- Delete the DFT Test package Data Flow task.
- Delete the AdventureWorks2008_cm connection.
- Delete the AdventureWorksDW2008_cm connection.

► **Task 2: Configure the deployment properties and build the package**

- Set the SSIS_proj8 properties.
 - Page: **Deployment Utility**
 - Property: **CreateDeploymentUtility**
 - Value: **True**
- Build the SSIS_proj8 project.

► **Task 3: View the deployment utility manifest**

- Verify the XML data references to the SendMail.dtsConfig file.
 - Folder: E:\MOD08\Labfiles\Starter\SSIS_sol8\SSIS_proj8\bin\Deployment
 - File: **SSIS_proj8.SSISDeploymentManifest**

Results: After this exercise, the new package has been prepared for deployment.

Exercise 3: Deploying a Package

The main tasks for this exercise are as follows:

1. Use the Package Installation Wizard to deploy the package.
2. Verify that the package has been deployed.
3. Verify that the configuration file has been deployed.
4. Run the package.

► Task 1: Use the Package Installation Wizard to deploy the package

- Start the package installation wizard.
 - Folder: E:\MOD08\Labfiles\Starter\SSIS_sol8\SSIS_proj8\bin\Deployment\
 - Package: SSIS_proj8.SSISDeploymentManifest
- Set the Deploy SSIS Packages page options.
 - Option: **SQL Server deployment**
 - Checkbox: **Validate packages after installation**
- Set the Specify Target SQL Server page options.
 - Server name: **(local)**
 - Authentication type: **Use Windows Authentication**
 - Package path: **/Data Collector**
 - Checkbox: **Rely on server storage for encryption**
- Set the Select Installation Folder page options.
 - Folder: C:\Program Files\Microsoft SQL Server\100\DTSPackages\SSIS_proj8
- Confirm the installation options.
- Confirm the Packages Validation page options.
- Finish the package installation wizard.

► **Task 2: Verify that the package has been deployed**

- Start the Server Management Studio and connect to the server.
- Connect to the integration services.
 - Server name: **localhost**
- Verify that the Employees package has been deployed to the local instance.

► **Task 3: Verify that the configuration file has been deployed**

- Verify that the configuration file has been deployed.
 - Folder: C:\Program Files\Microsoft SQL Server\100\DTSPackages\SSIS_proj8
 - File: SendMail.dtsConfig

► **Task 4: Run the package**

- Run the Employees package.
- Review the settings used to run the Employees package.
- View the execution progress.
- Examine the results and make sure that the validation completes successfully.
- Turn off 6235A-NY-SQL-01 and delete changes.

Results: After this exercise, the new package has been deployed and tested.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips
- Tools

Review Questions

1. What are package configurations?
2. What are the five package configuration formats?
3. What are the two types of package configurations provided by integration services?
4. What are property expressions?
5. What can the package configuration organizer dialog box be used for?
6. What is a deployment utility?

Common Issues Related to Configuring and Deploying Packages

Identify the causes for the following common issues related to configuring and deploying packages and fill in the troubleshooting tips. For answers, refer to relevant lessons in the module.

Issue	Troubleshooting tip
Bad Data	
Package Execution	
Package Errors	
Run-time Permissions Issues	
External Data Providers	

Tools

Tool	Use for	Where to find it
Microsoft SQL Server Management Studio	<ul style="list-style-type: none">Managing SQL server databases and tables	Start All Programs Microsoft SQL Server 2008
SQL Server Business Intelligence Development Studio	<ul style="list-style-type: none">Managing SQL server applications	Start All Programs Microsoft SQL Server 2008

Module 9

Managing and Securing Packages

Contents:

Lesson 1: Managing Packages	9-3
Lesson 2: Securing Packages	9-17
Lab: Managing and Securing Packages	9-26

Module Overview

- Managing Packages
- Securing Packages

Microsoft® SQL Server® Integration Services provides tools that make it easy to monitor and secure packages. These tools help you configure, schedule, secure, troubleshoot, import, and export packages. In this module, you will learn how to use these tools to monitor and secure packages.

Lesson 1

Managing Packages

- Managing Package Storage
- Importing and Exporting Packages
- Modifying Configuration Values at Runtime
- Scheduling Package Execution
- Monitoring Package Execution
- Troubleshooting Package Execution
- Backing Up and Restoring Packages

In order to better manage SQL Server packages, it is important to understand the many tools available to manage these packages. Sometimes it is necessary to schedule package execution. Sometimes it is necessary to backup, restore, import or export packages. Sometimes it is necessary to troubleshoot package execution. In this lesson, you will learn many techniques to manage these packages.

Managing Package Storage

Manage packages in the two storage folders

Root Folder	Notes
File System	• Lists packages that are saved to the file system
MSDB	• Lists packages that are saved to the msdb database

Integration Services in SQL Server Management Studio

Node	Notes
Running Packages	• Lists packages currently running on the server
Stored Packages	• Lists packages saved in the package store

Configure the folder location in IS service configuration

Add subfolders to the root folders or to other subfolders

Key Points

Packages can be managed in two storage folders.

- The File System folder lists packages that are saved to the file system.
- The MSDB folder lists packages that are saved to the msdb database.
- Packages that you save to msdb are stored in a table named sysssispackages.
- The Running Packages node lists the packages that are currently running on the server.
- The Stored Packages node lists the packages saved in the package store.

Question: Which technique might serve your organization better, storing packages in the file system or the msdb database?

Importing and Exporting Packages

The diagram consists of two main sections. The top section contains two items: 'SQL Server Management Studio' with an icon of a monitor and a database, and 'The dtutil command prompt utility' with an icon of a monitor showing 'C:/'. Below this is a blue box containing the text 'Packages can be imported and exported between:' followed by three bullet points: 'File system folders anywhere in the file system', 'Folders in the SSIS package store', and 'Between SQL Server instances'.

Packages can be imported and exported between:

- File system folders anywhere in the file system
- Folders in the SSIS package store
- Between SQL Server instances

Key Points

Integration Services supports two methods for importing and exporting packages: the built-in capabilities of SQL Server Management Studio and the dtutil command prompt utility.

- Using the import and export features, you can add packages to the file system, package store, or msdb database, and copy packages from one storage format to another.
- The dtutil command prompt utility enables you to copy, move, delete, or verify a package.

Question: In what scenarios might you employ these techniques for importing and exporting packages?

Demonstration: Importing a Package Using dtutil

In this demonstration, you will see how to import a package using dtutil by:

- Creating a folder structure for packages
- Viewing the data about the folders in the msdb database
- Importing the demographics package into a folder
- Viewing the data about the package in the msdb database

Question: How can the dtutil command prompt utility be used?

Modifying Configuration Values at Runtime

```
dtexec.exe /option [value] [/option [value]]...
```

Option	Description
/CommandFile <i>filespec</i>	The file specified in filespec is opened and options from the file are read until EOF
/ConfigFile <i>filespec</i>	Specifies a configuration file to extract values
/File <i>filespec</i>	Loads a package saved in the file system
/Set <i>propertyPath;value</i>	Overrides the configuration of a variable, property, or connection within a package
/VerifySigned	Causes Integration Services to check the digital signature of the package

Key Points

The dtexec utility provides access to all the package configuration and execution features, such as connections, properties, variables, logging, and progress indicators.

- **/Com[mandFile]** *filespec* specifies that during the command sourcing phase of the utility, the file specified in filespec is opened, options from the file are read until EOF is found in the file.
- **/Conf[igFile]** *filespec* specifies a configuration file to extract values from.
- **/F[ile]** *filespec* loads a package that is saved in the file system.
- **/Set** *propertyPath;value* overrides the configuration of a variable, property, container, log provider, Foreach enumerator, or connection within a package.
- **/VerifyS[igned]** causes Integration Services to check the digital signature of the package.

Scheduling Package Execution

SQL Server Agent is a Windows service that executes scheduled administrative tasks, which are called *jobs*

1 Connect to instance of a SQL Server database engine

2 Create a new SQL Server Agent job

3 Create a step in the new job

4 Create a schedule in the new job



Key Points

SQL Server Agent is a Microsoft Windows® service that executes scheduled administrative tasks, which are called jobs.

- SQL Server Agent must be active before local or multi-server jobs can run automatically.
- You can enhance the job by setting notification options, such as specifying an operator to send an e-mail message to when the job finishes, or adding alerts.

Demonstration: Scheduling Package Execution

In this demonstration, you will see how to schedule a package for execution by:

- Creating a SQL Server Agent job to run a package

Question: What other ways are there of scheduling packages besides the one shown here?

Monitoring Package Execution

Monitoring Tool	Where Found
Package Execution Progress	<ul style="list-style-type: none">Execute Package utility
Running Packages node	<ul style="list-style-type: none">Integration Services node
Start Jobs window	<ul style="list-style-type: none">SQL Server Agent
Job Activity Monitor window	<ul style="list-style-type: none">SQL Server Agent
Log File Viewer window	<ul style="list-style-type: none">Package
Performance Monitor	<ul style="list-style-type: none">Start Programs Administrative Tools Performance

Key Points

There are many tools and techniques available for monitoring package executions.

- When you use the Execute Package utility to run a package manually, the utility displays the Package Execution Progress window.
- The node lists each package as it runs. You can generate a report for all packages that are running. This report provides information such as the package name and when execution started.
- When you start a SQL Server Agent job manually, SQL Server Agent displays the Start Jobs window. The window provides you with the status of the job as it runs and indicates whether each action succeeds or fails.

- The window provides a list of SQL Server Agent jobs. You can view the current status of a job, a brief overview of the job, and whether the last execution was successful or whether it failed.
- For each package, you can open the viewer to view a history of each time the package ran. The window provides details about the job's outcome and about each step.

Question: Which of these tools have you used for monitoring package execution?

Demonstration: Monitoring Package Execution

In this demonstration, you will see how to monitor package execution by:

- Viewing job activity
- Viewing job history

Question: What information can you view in the Job Activity Monitor window?

Troubleshooting Package Execution

Techniques for troubleshooting package execution:

- Catch and handle package errors by using event handlers**
- Capture bad data by using error outputs**
- Track the steps of package execution by using logging**
- Log before and after calls to an external data provider**



Key Points

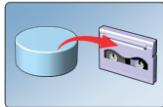
There are many techniques available for troubleshooting package execution.

- You can respond to the many events that are raised by the package and the objects in the package by using event handlers.
- In the event handler, you can use a Send Mail task to notify an administrator of the failure, use a Script task and custom logic to obtain system information for troubleshooting, or clean up temporary resources or incomplete output.
- You can use the error output available on many data flow components to direct rows that contain errors to a separate destination for later analysis.

- You can track much of what occurs in your running packages by enabling logging.
- Integration Services includes logging messages that you can use to troubleshoot a package's interaction with external data sources.

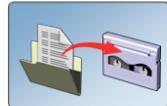
Question: Have you ever used any of these techniques for troubleshooting package execution?

Backing Up and Restoring Packages



Use the SQL Server back-up and restore features

Packages saved to msdb can be backed up and restored using the SQL Server backup and restore features



Use the file system back-up and restore plan

Packages saved to the file system should be included in the plan for backing up the file system of the server

The SQL Server back-up feature will not back up configuration files saved to the file system



Key Points

Backup and restore of packages can be managed with the SQL Server back-up feature or the file system back-up system.

- Packages saved to msdb can be backed up and restored using the SQL Server backup and restore features.
- The SQL Server back-up feature will not back up configuration files saved to the file system. You must back up these files separately.
- You should ensure that the package files, along with any configuration files, are part of the back-up and restore plan that is used to protect your server and network file systems.
- The Integration Services service configuration file, which has the default name MsDtsSrvr.ini.xml, lists the folders on the server that the service monitors. You should make sure these folders are backed up.

Demonstration: Backup and Restore Packages

In this demonstration, you will see how to perform backups by:

- Using the SQL Server backup tools

Question: What is the difference between the three different recovery models?

Lesson 2

Securing Packages

- Overview of Integration Services Security
- Package Protection Levels
- Package Roles
- Signing Packages

It is important to understand the security issues surrounding SQL Server packages. There are many ways to better maintain security of packages, such as using roles and digital certificates. In this lesson, you will learn how to better maintain package security.

Overview of Integration Services Security



Ensure package security in the development and production environments

Integration Services implements security by:

- Setting ProtectionLevel to specify if data should be encrypted
- Setting package properties to protect packages by encryption
- Controlling access using SQL Server database-level roles
- Securing the environment by protecting file location access
- Guaranteeing the integrity of packages by signing packages

Key Points

Security in SQL Server Integration Services is comprised of several layers that provide a rich and flexible security environment. Integration Services security combines the use of package level properties, SQL Server database roles, operating system permissions, and digital signatures.

Here are some best practices:

- Configure the appropriate protection level on the package.
- Digitally sign your packages to prevent unauthorized access.
- Use the appropriate folder permissions to restrict file access to authorized users only.
- Assign the appropriate database roles to packages deployed to SQL Server.
- Use the appropriate folder permissions to protect packages and configuration files deployed to the file system.

- Restrict access to computers running the SQL Server service.
- Set the **ProtectionLevel** property of the package to specify whether sensitive data should be protected by encryption, or whether the data should be removed before saving.
- Set the **ProtectionLevel** and **PackagePassword** properties of the package to protect packages by encrypting all or part of a package using passwords or user keys.
- Secure the operational environment by protecting file locations and limiting access to packages in SQL Server Management Studio.
- Guarantee the integrity of packages by signing packages with certificates.

Question: Have you ever employed any of these techniques for securing packages?

Package Protection Levels

Protection Level	Description
DontSaveSensitive	Suppresses sensitive information in the package when it is saved
EncryptAllWithPassword	Encrypts the whole package by using a password
EncryptAllWithUserKey	Encrypts the whole package by using a key based on the user profile
EncryptSensitiveWithPassword	Encrypts only the sensitive information in the package by using a password
EncryptSensitiveWithUserKey	Encrypts only the sensitive information in the package by using current user keys
ServerStorage	Protects the whole package using SQL Server database roles

Key Points

You can set the protection level of a SQL Server Integration Services package when you first develop it in Business Intelligence Development Studio.

- **Do not save sensitive (DontSaveSensitive)**

This protection level does not encrypt, but instead it prevents properties that are marked sensitive from being saved with the package, and therefore makes the sensitive data unavailable to other users.

- **Encrypt all with password (EncryptAllWithPassword)**

The package is encrypted by using a password that the user supplies when the package is created or exported.

- **Encrypt all with user key (EncryptAllWithUserKey)**

Only the same user using the same profile can load the package.

- **Encrypt sensitive with password (EncryptSensitiveWithPassword)**

Sensitive data is saved as a part of the package, but that data is encrypted by using a password that the current user supplies when the package is created or exported.

- **Encrypt sensitive with user key (EncryptSensitiveWithUserKey)**

Only the same user using the same profile can load the package.

- **Rely on server storage for encryption (ServerStorage)**

This option is supported only when a package is saved to the SQL Server msdb database.

Question: Which package protection level might prove to be most useful to your organization?

Package Roles

	Add user-defined roles assigned to the msdb database
	Specify reader and writer roles for each package
	Integration services includes the three fixed database-level roles
Role	Member Privileges
db_dtsadmin	<ul style="list-style-type: none">Members have full access to all packages
db_dtsltduser	<ul style="list-style-type: none">Enumerate all, but only manage packages they own
db_dtsoperator	<ul style="list-style-type: none">Members can view, execute, and export, but they cannot import or modify packages

Key Points

You can assign roles to packages to control access to those packages.

- SQL Server Integration Services includes the three fixed database-level roles—db_dtsadmin, db_dtsltduser, and db_dtsoperator—for controlling access to packages.
- Roles can be implemented only on packages that are saved to the msdb database in SQL Server.
- The roles listed in the Package Role dialog box are the current database roles of the msdb system database. If no roles are selected, the default Integration Services roles apply. By default, the reader role includes db_dtsadmin, db_dtsoperator, and the user who created the package.

- **db_dtsadmin** Members of this role have full access to all packages.
- **db_dtsltduser** Members of this role can enumerate all packages, but only manage the packages they own.
- **db_dtsoperator** Members of this role can view, execute, and export all packages on the SQL Server instance, but they cannot import or modify packages.

Signing Packages



Use digital signing



Use dtutil

Important:

- When configured to check the signature of the package, Integration Services only checks whether the digital signature is present, is valid, and is from a trusted source. Integration Services does not check whether the package has been changed.

Key Points

An Integration Services package can be signed with a digital certificate to identify its source. A digital signature prevents an unauthorized user from tampering with or running a package.

- To sign a package by using the Digital Signing dialog box, open the package in SSIS Designer, and on the SSIS menu, click **Digital Signing**. You must then select a certificate with which to sign the package.
- To use the **dtutil** command prompt utility to sign a package, run the utility with the **/Si** option, specifying the path to the package and the hexadecimal thumbprint of the certificate.

Demonstration: Securing a Package

In this demonstration, you will see how to secure a package by:

- Opening an Integration Services solution
- Encrypting a package with a user key
- Importing the package
- Configuring the package roles for the package
- Running the package

Question: What does the EncryptAllWithUserKey ProtectionLevel do?

Lab: Managing and Securing Packages

- Exercise 1: Importing a Package
- Exercise 2: Configuring, Executing, and Monitoring a Package
- Exercise 3: Scheduling a Package
- Exercise 4: Securing a Package

Logon information

Virtual machine	NY-SQL-01
User name	Student
Password	Pa\$\$w0rd

Estimated time: 90 minutes

Exercise 1: Importing a Package

Scenario

The AdventureWorks IT Director has decided to expand their SQL Server package deployment infrastructure. The IT Director has requested that a number of new features to be added to their current package deployment infrastructure to make it more flexible, powerful, and reliable.

You have developed an Integration Services project that includes two packages. The Demographics package retrieves demographic sales data from the AdventureWorks database and inserts that data into the HumanResources database.

The Employees package retrieves data from a text file and inserts that data into the HumanResources database. You now plan to import the packages into the local instance of Integration Services. You will run, monitor, and schedule the Demographics package, and you will configure security on the Employees package.

In this exercise, you must import a package.

The main tasks for this exercise are as follows:

1. Configure the lab environment.
2. Create a folder structure for packages.
3. View the data about the folders in the msdb database.
4. Import the package into a folder.
5. View the data about the folders in the msdb database.

► **Task 1: Configure the lab environment**

- Start 6235A-NY-SQL-01 and logon as **Student** using the password **Pa\$\$w0rd**.
- Run the setup code.
 - Name: **Setup.cmd**
 - Location: **E:\MOD09\Labfiles\Starter**
- Create the folder: **C:\MOD09\Labfiles**.
- Copy the **SendMail** task file to the C: drive.
 - File name: **SendMail.dtsConfig**
 - Source folder: **E:\MOD09\Labfiles\Starter\Employees**
 - Destination folder: **C:\MOD09\Labfiles**
- Start the Server Management Studio and connect to the server.
- Connect to the integration services.
 - Server name: **localhost**

► **Task 2: Create a folder structure for packages**

- Create the folders in Object Explorer.
 - Folder: **Stored Packages | MSDB | HumanResources**
 - Folder: **Stored Packages | MSDB | Sales**
 - Folder: **Stored Packages | MSDB | HumanResources | Employees**

► **Task 3: View the data about the folders in the msdb database**

- Execute the query.
 - Available Database: **msdb**
 - SQL statement: **SELECT * FROM dbo.syssispackagefolders**

► **Task 4: Import the package into a folder**

- Import the package into a folder.
 - Object Explorer folder: **Stored Packages | MSDB | Sales**
 - Package location: **File System**
 - Package path folder: **E:\MOD09\Labfiles\Starter\SSIS_sol9\SSIS_proj9\bin**
 - Package: **Demographics.dtsx**

► **Task 5: View the data about the package in the msdb database**

- Execute the query.
 - Available Database: **msdb**
 - SQL statement: **SELECT * FROM dbo.syssispackages**

Results: After this exercise, the package has been imported and tested.

Exercise 2: Configuring, Executing, and Monitoring a Package

Scenario

The IT Director has requested a number of new features to be added to the AdventureWorks package infrastructure. The IT Director has requested to add configuration files to the Demographics package, expand logging for the package, and better manage the package events handlers.

In this exercise, you must configure, execute and monitor a package.

The main tasks for this exercise are as follows:

1. Add a configuration file to the package execution.
2. Configure the package to fail on validation warnings.
3. Configure console logging to log the source name and message.
4. Disable event handlers.
5. Run the package.
6. View the package execution progress.
7. Verify that an email message was sent.

► Task 1: Add a configuration file to the package execution

- Add a configuration file to the package execution.
 - Object Explorer folder: **Stored Packages | MSDB | Sales**
 - Package: **Demographics**
 - Configuration file folder: **E:\MOD09\Labfiles\Starter\Employees**
 - Configuration file: **SendMail.dtsConfig**

► **Task 2: Configure the package to fail on validation warnings**

- Configure the package to fail on validation warnings.
 - Page: **Execution Options**
 - Select the checkbox: **Fail the package on validation warnings**
 - Clear the checkbox: **Validate package without executing**
 - Clear the checkbox: **Enable package checkpoints**
 - Maximum concurrent executables: **-1**

► **Task 3: Configure console logging to log the source name and message**

- Configure console logging to log the source name and message.
 - Page: **Reporting**
 - Section: **Console logging**
 - Select the checkbox: **Source name**
 - Select the checkbox: **Message**
 - Section: **Console events**
 - Select the checkbox: **Verbose**

► **Task 4: Disable event handlers**

- Disable event handlers.
 - Page: **Set Values**
 - Property Path field: **\Package.Properties[DisableEventHandlers]**
 - Value field: **True**

► **Task 5: Run the package**

- Click **Execute**.
 - Folder: **Running Packages**

► **Task 6: View the package execution progress**

- View the package execution progress.
- Verify that all validations completed successfully.

► **Task 7: Verify that an email message was sent**

- Verify that the SMT Send e-mail on success Send Mail task has sent an e-mail message.
 - Mail folder: C:\inetpub\mailroot\Queue

Results: After this exercise, the package has been configured, executed, and then monitored.

Exercise 3: Scheduling a Package

Scenario

The IT Director has found that package deployment has been interrupting the enterprise's servers and occupying too much of the network bandwidth. The IT Director has requested that package deployments to be moved to evenings, after the organization's normal work day has ended. Instead of working overtime, your manager has requested that you create a server agent job to automatically deploy packages after hours.

In this exercise, you must schedule a package.

The main tasks for this exercise are as follows:

1. Create a SQL Server Agent job to run a package.
2. View job activity.
3. View job history.

► Task 1: Create a SQL Server Agent job to run a package

- Create a SQL Server Agent job in Object Explorer to run a package.
 - Node: **SQL Server Agent | Jobs**
 - Page: **General**
 - Name: **Run Demographics**
 - Owner field: **NY-SQL-01\Administrator**
 - Category list: **[Uncategorized (Local)]**
 - Select the checkbox: **Enabled**
- Create a new job step.
 - Page: **Steps**
 - Step name: **Run package**
 - Type: **SQL Server Integration Services Package**
 - Package source: **SSIS Package Store**
 - Server: **NY-SQL-01**
 - Log on to the server: **Use Windows Authentication**

- Add the package.
 - Folder: \MSDB\Sales
 - Package: **Demographics**
- Add the configuration.
 - Tab: **Configurations**
 - Folder: C:\MOD09\Labfiles
 - File: **SendMail.dtsConfig**
- Create a new job schedule.
 - Page: **Schedules**
 - Name: **Run once**
 - Schedule type: **One time**
 - Time: *Set for 2 minutes into the future*

► **Task 2: View job activity**

- View the details about the Run Demographics job using the job activity monitor.
- Refresh until the Status is shown as **Executing**.
- Refresh until the Status is shown as **Idle**.
- Close the job activity monitor.

► **Task 3: View job history**

- Using the Object Explorer, view the job history for the Run Demographics job.
- Close the Log File Viewer.

Results: After this exercise, the package has been scheduled for execution.

Exercise 4: Securing a Package

Scenario

The IT Director has requested to secure the package deployment infrastructure to increase network security. Your manager has decided that the best way of securing packages is by encrypting all packages with a user key.

In this exercise, you must secure a package.

The main tasks for this exercise are as follows:

1. Open an Integration Services solution.
2. Encrypt a package with a user key.
3. Import the package.
4. Configure the package roles for the package.
5. Run the package.

► Task 1: Open an Integration Services solution

- Start SQL Server Business Intelligence Development Studio.
- Close the default Start page.
- Open and examine package solution.
 - Folder: E:\MOD09\Labfiles\Starter\SSIS_sol9
 - Solution: SSIS_sol9.sln
 - Package: Employees.dtsx

► Task 2: Encrypt a package with a user key

- Set the control flow design surface properties to encrypt a package with a user key.
 - ProtectionLevel: **EncryptAllWithUserKey**
- Save All.
- Build SSIS_proj9.

► **Task 3: Import the package**

- Import the package into a folder.
 - Object Explorer folder: **Stored Packages | MSDB | HumanResources | Employees**
 - Package location: **File System**
 - Package path folder: **E:\MOD09\Labfiles\Starter\SSIS_sol9\SSIS_proj9\bin**
 - Package: **Employees.dtsx**
- Verify that the Employees package has been added.
 - Object Explorer folder: **Stored Packages | MSDB | HumanResources | Employees**

► **Task 4: Configure the package roles for the package**

- Configure the package roles for the package.
 - Package: **Employees**
 - Reader Role: <Default: db_dtsadmin, db_dtsoperator, creator of the package>
 - Writer Role: <Default: db_dtsadmin, creator of the package>

► **Task 5: Run the package**

- Run the **Employees** package.
- View the execution progress in the **Package Execution Progress** window.
- Close the **Package Execution Progress** dialog box.
- Close the **Execute Package Utility** dialog box.
- Turn off **6235A-NY-SQL-01** and delete changes.

Results: After this exercise, the package has been secured.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips
- Tools

Review Questions

1. What are the two folders for packages storage?
2. Name some of the primary options that can be used to modifying configuration values at runtime?
3. Name some of the methods for monitoring package execution.
4. What are the two package location considerations for backing up and restoring packages?
5. What are some of the techniques for troubleshooting package execution?
6. What are the three fixed database-level roles included in SQL Server Integration Services?
7. What are the two methods you can use to sign a package with Integration Services?

Common Issues related to managing and securing packages

Identify the causes for the following common issues related to managing and securing packages and fill in the troubleshooting tips. For answers, refer to relevant lessons in the module.

Issue	Troubleshooting tip
Bad Data	
Package Execution	
Package Errors	
Run-time Permissions Issues	

Tools

Tool	Use for	Where to find it
Microsoft SQL Server Management Studio	<ul style="list-style-type: none">Managing SQL server databases and tables	Start All Programs Microsoft SQL Server 2008
SQL Server Business Intelligence Development Studio	<ul style="list-style-type: none">Managing SQL server applications	Start All Programs Microsoft SQL Server 2008

Course Evaluation



Your evaluation of this course will help Microsoft understand the quality of your learning experience.

Please work with your training provider to access the course evaluation form.

Microsoft will keep your answers to this survey private and confidential and will use your responses to improve your future learning experience. Your open and honest feedback is valuable and appreciated.

Module 1: Introduction to SQL Server Integration Services

Lab: Using SQL Server Integration Services

Exercise 1: Using the Import and Export Wizard

- ▶ **Task 1: Open SQL Server Management Studio and connect to the database engine on NY-SQL-01**
 1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
 2. Log on to NY-SQL-01 as **Student** using the password **Pa\$\$w0rd**.
 3. Click **Start**, click **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
 4. In the **Connect to Server** dialog box, click **Connect**.

- ▶ **Task 2: Use the SQL Server Import and Export Wizard to export the currency data**
 1. In Object Explorer, expand the **Databases** node.
 2. Right-click **AdventureWorks2008**, point to **Tasks**, and then click **Export Data**.

► **Task 3: Choose a data source**

1. In the **SQL Server Import and Export Wizard**, click **Next**.
2. On the Choose a Data Source page, specify the values in the following table, and then click **Next**:

Property	Value
Data source	SQL Native Client 10.0
Server Name	NY-SQL-01
Authentication	Use Windows Authentication
Database	AdventureWorks2008

► **Task 4: Choose a destination**

1. On the Choose a Destination page, select **Flat File Destination** from the **Destination** list.
2. In the **File name** box, type **E:\Mod01\Labfiles\Currency.txt**.
3. Select the **Column names in the first data row** check box.
4. Use the default settings for the other options, and then click **Next**.

► **Task 5: Write a Transact-SQL statement to extract currency data**

1. On the Specify Table Copy or Query page, select **Write a query to specify the data to transfer**, and then click **Next**.
2. On the Provide a Source Query page, type the following SQL statement:

```
SELECT CurrencyCode, Name FROM Sales.Currency
```

3. Click **Parse**, and then when the syntax is validated, click **OK**.
4. Click **Next**.

► **Task 6: Configure the flat file destination**

1. On the Configure Flat File Destination page, specify the values in the following table:

Property	Value
Source query	[Query]
Row delimiter	{CR}{LF}
Column delimiter	Comma {,}

2. Click **Preview**. After you have viewed the source data, click **OK**.
3. Click **Next**.

► **Task 7: Create an Integration Services package**

1. On the Save and Run Package page, clear the **Run immediately** check box, select the **Save SSIS Package** check box, and then ensure that **SQL Server** is selected.
2. In the **Package protection level** drop down, click **Rely on server storage and roles for access control**.
3. Click **Next**.

► **Task 8: Save the Integration Services package**

1. On the Save SSIS Package page, type **CurrencyExport** in the **Name** box.
2. In the **Description** box, type **Export currency data**.
3. Ensure that **Server name** is set to **NY-SQL-01**, and that **Use Windows Authentication** is selected, and then click **Next**.
4. On the Complete the Wizard page, review the information, and then click **Finish**.
5. Review the details about the actions performed by the wizard, and then click **Close**.

Results: After this exercise, you should have created a package capable of exporting the Currency table from the database. The package should now be stored in the msdb database as **CurrencyExport**.

Exercise 2: Running an Integration Services Package

► **Task 1: Connect to Integration Services**

1. In Object Explorer, click **Connect**, and then click **Integration Services**.
2. In the **Connect to Server** dialog box, ensure that **NY-SQL-01** is selected in the **Server name** box, and then click **Connect**.

► **Task 2: Launch the Execute Package Utility**

1. In Object Explorer, expand the **Stored Packages** node.
2. Expand the **MSDB** node.
3. Right-click the **CurrencyExport** package, and then click **Run Package**.
4. In the Execute Package Utility, review the settings on the General page and Connection Managers page.

► **Task 3: Run the Currency package**

1. Use the default settings for all options in the Execute Package Utility, and then click **Execute**.
2. View the execution progress in the Package Execution Progress window, and then click **Close**.
3. Click **Close** to close the Execute Package utility.
4. Minimize SQL Server Management Studio.

► **Task 4: View the Currency.txt file**

1. Click **Start**, and then click **Computer**.
2. In Windows Explorer, navigate to **E:\Mod01\Labfiles**.
3. Double-click the **Currency.txt** file to open the file in Notepad.
4. Verify that the file contains the currency codes and names.
5. On the **File** menu, click **Exit**.

Results: After this exercise, you should have successfully exported the contents of the currency table to the file **E:\Mod01\Labfiles\Currency.txt**.

Exercise 3: Import Data Using the Import and Export Wizard

► Task 1: Import data using the Import and Export Wizard

1. In Object Explorer, right-click AdventureWorks2008, point to Tasks, and then click **Import Data**.
2. On the Welcome to SQL Server Import and Export Wizard page, click **Next**.
3. On the Choose a Data Source page, specify the values in the following table, and then click **Next** twice:

Property	Value
Data source	Flat File Source
File Name	E:\Mod01\Labfiles\Currency.txt
Locale	English (United States)
Code Page	1252 (ANSI - Latin I)
Format	Delimited
Text qualifier	<none>
Header row delimiter	{CR}/{LF}
Header rows to Skip	0
Column names in the first data row	Selected

4. In the Choose a Destination page, accept the default values and click **Next**.
5. In the Select Source Table and Views page, accept the default values and click **Next**.
6. In the **Save and Run Package** page, click **Next**.
7. Click **Finish**.
8. Click **Close**.

Results: After this exercise, you should have successfully imported the contents of **E:\Mod01\Labfiles\Currency.txt** to the AdventureWorks database.

Exercise 4: Managing Packages

► **Task 1: Export a package to the file system using SQL Server Management Studio**

1. In Object Explorer, under **NY-SQL-01 (Integration Services 10.0.1300 – NY-SQL-01)**, right-click **CurrencyExport**, and then click **Export Package**.
2. In the **Export Package - \MSDB\CurrencyExport** dialog box, specify the values in the following table, and then click **OK**:

Property	Value
Package Location	File System
Package Path	E:\Mod01\Labfiles\CurrencyExport1
Protection Level	Encrypt sensitive data with user key

3. Browse to **E:\Mod01\Labfiles** and confirm that **CurrencyExport1.dtsx** was created.

► **Task 2: Edit a package**

1. In Windows Explorer, right-click **CurrencyExport1.dtsx**, and then click **Edit**.
2. In the Microsoft Visual Studio window, click the **Data Flow** tab.
3. Right-click **Source – Query** and then click **Edit**.
4. In the **OLE DB Source Editor** dialog box, select the contents of the **SQL command** text field and type the following query:

```
SELECT Firstname, Lastname  
FROM Person.Person  
WHERE BusinessEntityID < 20
```

5. Click **Parse Query**.
6. When the dialogue appears stating The SQL statement was successfully parsed, click **OK**.
7. Click **Columns**.
8. Confirm that Columns **Firstname** and **Lastname** are checked.

9. Click **OK**.
10. Right-click **Destination – Currency**, and then click **Edit**.
11. In the **Restore Invalid Column References Editor** dialog box, in the **Available Columns** column, for the first row, select **Source - Query.FirstName**.
12. In the **Available Columns** column, for the second row, select **Source - Query.LastName**.
13. Click **OK**.
14. In the **Connection Manager** field, right-click **DestinationConnectionFlatFile** and then click **Edit**.
15. In the **Flat File Connection Manager Editor** dialog box, in the **File name** field, type **E:\MOD01\Labfiles\Contacts.txt**.
16. Click **Advanced**.
17. Click **CurrencyCode**.
18. In the right column, select **CurrencyCode**, and then type **FirstName**.
19. In the center column, click **Name**.
20. In the right column, select **Name**, and then type **LastName**.
21. Click **OK**.
22. Right-click **Destination – Currency_txt**, and then click **Edit**.
23. If a dialog box appears, click **Yes**.
24. Click **Mappings**.
25. In the **Flat File Destination Editor** dialog box, click **OK**.

► **Task 3: Save the package**

1. Click **File**, and then click **Save Copy of CurrencyExport.dtsx As**.
2. In the **Save Copy of Package** dialog in the **Package Location** drop down, click **File System**.
3. In the **Package path** dialog, type **E:\MOD01\Labfiles>ContactExport.dtsx**.
4. Click **OK**.

► **Task 4: Import a package using DTUTIL**

1. Click **Start**, and then click **Command Prompt**.
2. At the command prompt, type **dtutil /FILE E:\MOD01\LabFiles\ContactExport.dtsx /COPY SQL;ContactExport**, and then press **ENTER**.
3. When the command prompt window displays **The operation was completed successfully**, minimize the window.
4. Restore Microsoft SQL Server Management Studio.
5. Under the **NY-SQL-01 (Integration Services 10.0.1300 -NY-SQL-01)**, right-click **Stored Packages**, and then click **Refresh**.
6. Expand the **MSDB** node.
7. Confirm that **ContactExport** is present.

► **Task 5: Run a package using DTExec**

1. Restore the command prompt window.
2. At the command prompt, type **dtexec /SQL contactexport**, and then press **ENTER**.
3. When the command prompt window displays **DTExec: The package execution returned DTSSER_SUCCESS (0)**, minimize the command prompt window.
4. Open Windows Explorer and browse to **E:\MOD01\LabFiles**.
5. Double click **Contacts.txt**.
6. Confirm the contents of the file.
7. Turn off Virtual Machine and delete changes.

Results: After this exercise, you should have modified an existing package, imported it into the SQL msdb database, and executed it.

Module 2: Developing Integration Services Solutions

Lab: Developing Integration Services Solutions

Exercise 1: Creating an Integration Services Project

► **Task 1: Create an integration services project**

1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
2. Log on to **NY-SQL-01** as **Student** using the password **Pa\$\$w0rd**.
3. Click **Start**, click **All Programs**, click **Microsoft SQL Server 2008**, and click **SQL Server Business Intelligence Development Studio**.
4. Right-click the **Start Page** tab, and then click **Close**.
5. On the **File** menu, point to **New**, and then click **Project**.
6. In the **New Project** dialog box, in the **Templates** pane, click **Integration Services Project**.
7. In the **Name** box, type **SSIS_proj2**.
8. In the **Location** box, type **E:\Mod02\Labfiles**.
9. In the **Solution Name** box, type **SSIS_sol2**.
10. Verify that the **Create directory for solution** check box is checked.
11. Click **OK**.

► **Task 2: Rename the package and the package object**

1. In Solution Explorer, right-click **Package.dtsx** (beneath the **SSIS Packages** node), and click **Rename**.
2. In the file name box, type **Products.dtsx**, and then press ENTER.
3. When prompted to rename the package object, click **Yes**.

► **Task 3: View the tools available for the control flow and the data flow**

1. Verify that the **Control Flow** tab is selected.
2. Review the list of tasks and containers available for the control flow in the Toolbox.

Tip: By default, the Toolbox is located on the left side of the screen. If the Toolbox is not visible, on the **View** menu, click **Toolbox**.

3. Click the **Data Flow** tab.
4. Review the list of sources, transformations, and destinations available to the data flow.

► **Task 4: View the file structure where the project and solution files are stored**

1. Click **Start**, and then click **Computer**.
2. In Microsoft® Windows® Explorer, view the **E:\Mod02\Labfiles\SSIS_sol2** folder, and review the subfolders and files.
3. Close Windows Explorer.

Results: After this exercise, you should have created and saved an SSIS package project.

Exercise 2: Implementing a Package

► Task 1: Create a data source

1. In Business Intelligence Development Studio, right-click the **Data Sources** node in Solution Explorer, and then click **New Data Source**.
2. On the Welcome to the Data Source Wizard page, click **Next**.
3. On the Select how to define the connection page, click **New**.
4. In the **Connection Manager** dialog box, in the **Server name** box, type **NY-SQL-01**.
5. Verify that the **Use Windows Authentication** option is selected.
6. In the **Select or enter a database name** list, click **AdventureWorks2008**.
7. Click **Test Connection**, and click **OK** when the connection is verified.
8. Click **OK** to close the **Connection Manager** dialog box.
9. On the Select how to define the connection page of the **Data Source Wizard**, click **Next**.
10. On the Completing the Wizard page, in the **Data source name** box, type **AdventureWorks2008.ds**.
11. Click **Finish**.

► Task 2: Create a data source view

1. Right-click the **Data Source Views** node in Solution Explorer, and then click **New Data Source View**.
2. On the Welcome to the Data Source View Wizard page, click **Next**.
3. On the Select a Data Source page, verify that **AdventureWorks2008.ds** is selected in the **Relational data sources** pane, and then click **Next**.
4. On the Select Tables and Views page, in the **Available objects** pane, click **Product (Production)**, and then click the single right arrow (>) to move the object into the **Included objects** pane.
5. Click **Next**.
6. On the **Completing the Wizard** page, in the **Name** box, type **Products_dsv**, and then click **Finish**.

7. In SSIS Designer, review the **Products_dsv** design surface.
8. Expand the **Product** node in the **Tables** pane to view the columns in this table.
9. In the **Tables** pane, right-click **Product** node, and then click **Explore Data**.

Note: It will take some time for the Office Web Components to install.

10. Review the sample data in the **Table**, **PivotTable**, **Chart**, and **Pivot Chart** tabs.

Tip: The Pivot Table and Pivot Chart tabs are more useful when the data source view contains multiple tables. You can use them to view aggregated data, broken down by multiple dimensions.

11. Close the Explore Product Table window, and then close the **Products_dsv.dsv** [Design] window.

► **Task 3: Create a connection manager**

1. Open the **Products.dtsx** package in the package designer if it is not already open, and click the **Control Flow** tab.
2. Right-click the **Connection Managers** pane, and then click **New Connection From Data Source**.
3. In the **Select Data Source** dialog box, verify that **AdventureWorks2008.ds** is selected in the **Available Data Sources** pane, and then click **OK**.
4. In the **Connection Managers** pane, right-click **AdventureWorks2008.ds**, and then click **Rename**.
5. In the connection manager name box, type **AdventureWorks2008_cm**, and then press **ENTER**.

► **Task 4: Create a variable to hold the product count**

1. On the SSIS menu, click **Variables**.
2. In the **Variables** pane, click **Add Variable**.
3. In the row for the new variable, in the **Name** box, type **ProductCount**, and then press ENTER.
4. Ensure that the variable properties are configured with the following values:
 - **Scope:** Products
 - **Data type:** Int32
 - **Value:** 0
5. Click the **Close** icon to close the **Variables** pane.

► **Task 5: Define the control flow to support a data flow operation**

1. Drag a **Data Flow Task** from the Toolbox to the control flow design surface.
2. Right-click the **Data Flow Task**, and then click **Properties**.
3. In the **Properties** pane, in the **Name** box, type **DFT Retrieve product data**.
4. In the **Description** box, type **Retrieve product data from AdventureWorks2008 database**.
5. On the toolbar, click the **Save All** button.

► **Task 6: Add a script to the control flow to display the product count in a message box**

1. Drag a **Script Task** from the Toolbox to the control flow design surface.
2. Click the **Data Flow** task, and then drag the precedence constraint from the **Data Flow** task to the **Script** task.
3. Double-click the **Script** task.
4. On the General page of the **Script Task Editor** dialog box, in the **Name** box, type **SCR Display product count**.
5. In the **Description** box, type **Retrieve product count from ProductCount variable and display in message**.

6. On the Script page of the **Script Task Editor** dialog box, for the **ReadOnlyVariables** box, click the ellipses (...).
7. In the **Select Variables** dialog box, select **User::ProductCount**.
8. Click **OK**.
9. Click **Edit Script**.
10. In the Microsoft Visual Studio for Applications window, in the Project Explorer pane, double-click **ScriptMain.cs**.
11. Select the **Main()** subroutine on the right hand drop down box. The **Main()** subroutine begins with the following line of code:

```
public void Main()
```

12. Insert a line after `// TODO: Add your code here` and add the following code:

```
string MyVar;
MyVar = Dts.Variables["ProductCount"].Value.ToString();
MessageBox.Show(MyVar);
```

Your Main() subroutine should now look like the following code:

```
Public Void Main()
{
    string MyVar;
    MyVar = Dts.Variables["ProductCount"].Value.ToString();
    MessageBox.Show(MyVar);
    Dts.TaskResult = Dts.Results.Success
}
```

13. Press ALT+F4 to close the Microsoft Visual Studio for Applications window.
14. Click **OK** to close the **Script Task Editor** dialog box.
15. On the toolbar, click the **Save All** button.

► **Task 7: Add an OLE DB Source to the data flow to extract the product data**

1. Double-click the **Data Flow** task to go to the data flow design surface.
2. Drag an **OLE DB Source** from the Toolbox to the design surface.
3. Right-click the source, and click **Properties**.
4. In the **Properties** pane, in the **Name** box, type **OLE Retrieve product data**.
5. In the **Description** box, type **Retrieve data from Product table in AdventureWorks2008**.
6. Double-click the **OLE DB source**.
7. On the Connection Manager page of the **OLE DB Source Editor** dialog box, click **Products_dsv** in the OLE DB connection manager list, and then click **OK**.
8. Verify that **Table or view** is selected in the **Data access mode** list.
9. In the Name of the **table or the view** list, click **[Production].[Product]**.
10. Click **Preview** to view the source data, and then click **Close**.
11. Click **OK** to close the **OLE DB Source Editor** dialog box.
12. Click the **Save All** button on the toolbar.

► **Task 8: Add a Row Count transformation to the data flow to update the ProductCount variable**

1. Drag a **Row Count** transformation from the Toolbox to the design surface.
2. Click **OLE Retrieve product data** and connect the green data flow path to the **Row Count** transformation.
3. Double-click the **Row Count** transformation.
4. In the **Advanced Editor for Row Count** dialog box, in the **Name** box, type **CNT Retrieve row count**.
5. In the **Description** box, type **Add row count to the ProductCount variable**.

6. In the **VariableName** box, select **User::ProductCount**.
 7. View the default settings for the other properties, and then click **OK**.
 8. Click the **Save All** button on the toolbar.
- **Task 9: Add a Flat File destination to the data flow to insert product data into the Products.txt file**
1. Drag a **Flat File Destination** from the Toolbox to the design surface.
 2. Click **CNT Retrieve Row Count** and connect the data flow path from the **CNT Retrieve Row Count** transformation to the **Flat File destination**.
 3. Right-click the **Flat File destination**, and then click **Properties**.
 4. In the **Properties** pane, in the **Name** box, type **FFD Load product data**.
 5. In the **Description** box, type **Add data to the Products.txt file**.
 6. Double-click the **Flat File** destination.
 7. In the **Flat File Destination Editor** dialog box, click **New**.
 8. In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.
 9. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **Products_cm**.
 10. In the **Description** box, type **Connects to the Products.txt file**.
 11. In the **File name** box, type **E:\Mod02\Labfiles\Products.txt**.
 12. View the default settings for all other options, and click **OK**.
 13. In the **Flat File Destination Editor** dialog box, verify the column mappings on the **Mappings** page, and then click **OK**.
 14. Click the **Save All** button on the toolbar.

► **Task 10: Create an SMTP connection manager**

1. Right-click the **Connection Managers** pane, and then click **New Connection**.
2. In the **Add SSIS Connection Manager** dialog box, click **SMTP**, and then click **Add**.
3. In the **SMTP Connection Manager Editor** dialog box, in the **Name** box, type **LocalSmtp_cm**.
4. In the **Description** box, type **Connect to local SMTP server**.
5. In the **SMTP server** box, type **localhost**.
6. Verify that the **Use Windows Authentication** check box and the **Enable Secure Sockets Layer (SSL)** check box are cleared, and then click **OK**.

► **Task 11: Create an event handler**

1. Click the **Event Handlers** tab.
2. In the **Executable** list, select the **DFT Retrieve product data** node, and then click **OK**.
3. In the **Event handler** list, select **OnPostValidate**.
4. Click the hyperlink on the event handler design surface to create the event handler.

► **Task 12: Configure the event handler to send an e-mail message after the Products.txt file is created**

1. Drag a **Send Mail Task** from the **Toolbox** to the event handler design surface.
2. Double-click the task.
3. On the **General** page of the **Send Mail Task Editor** dialog box, in the **Name** box, type **SMTP Send notification**.
4. In the **Description** box, type **Send e-mail upon completion of the data load**.
5. On the **Mail** page of the **Send Mail Task Editor** dialog box, select **LocalSmtp_cm** from the **SmtpConnection** list.
6. In the **From** box, type **sqlserver@adventure-works.com**.
7. In the **To** box, type **administrator@NY-SQL-01.com**.

8. In the **Subject** box, type **Products updated**.
9. In the **MessageSource** box, type **The Products.txt file has been updated**.
10. Verify that **Direct Input** is selected in the **MessageSourceType** box and **Normal** is selected in the **Priority** box, and then click **OK**.
11. Click the **Save All** button on the toolbar.

► **Task 13: Add annotations to the Products package**

1. Click the **Control Flow** tab, and then click the design surface near the two elements.
2. Type a description that provides an overview of the control flow. When you begin typing, SSIS Designer automatically creates a text box that you can later resize and move. To insert a break at the end of a line, press CTRL+ENTER.
3. Click the **Data Flow** tab.
4. Click the design surface next to the **OLE DB** source, and type a description of the extraction operation.
5. Click the design surface next to the **Row Count** transformation, and type a description of the row count operation.
6. Click the design surface next to the **Flat File** destination, and type a description of the load operation.
7. Click the **Event Handlers** tab.
8. Click the design surface next to the **Send Mail** task, and type a description of the event handler operation.
9. Click the **Save All** button on the toolbar.

Results: After this exercise, you should have added the objects necessary to create the transformation and creation of a flat file from the products table. The SSIS package will also be able to display the row count and send an email upon project completion.

Exercise 3: Building and Running an Integration Services Project

► Task 1: Build the Integration Services project

1. On the **Build** menu, click **Build SSIS_proj2**.
2. Look for the **Build succeeded** message in the bottom left corner of SSIS Designer.
3. Use Windows Explorer to view the **E:\Mod02\Labfiles\SSIS_sol2\SSIS_proj2\bin** folder, and verify that SSIS Designer copied the **Products.dtsx** file into the folder.

► Task 2: Run the Products package

1. In Business Intelligence Development Studio, click the **Control Flow** tab.
2. Click **Start Debugging** on the toolbar.
3. In the dialog that appears, click **OK**.
4. Verify that each component turns green.

Note: The **DFT Retrieve Product Data** object will display a validation error.

Tip: You can use the scroll bars to view the Data Flow surface while debugging. You can also close the additional windows opened by the SSIS compiler when you built the package to make it easier to view the design surface.

5. Compare the row counts associated with the data flow paths with the row counts displayed in the **Script Task** message box.

► **Task 3: Stop running the package, and view the execution results**

1. Click **Stop Debugging** on the toolbar.
2. Click the **Execution Results** tab, and review the event information.

► **Task 4: Verify package execution**

1. Use Windows Explorer to view the E:\Mod02\Labfiles folder, and verify that Integration Services added the **Products.txt** file to the folder.
2. Double-click the **Products.txt** file to open the file in Notepad.
3. Verify that the file contains the product information, and then close Notepad.
4. Turn off virtual machine and delete changes.

Results: After this exercise, you should have run the package, creating the Products.txt file. The number of rows will have been displayed in a dialog box. Using the execution results tab, you will have identified problems with the SMTP server.

Module 3: Implementing Control Flow

Lab: Implementing Control Flow

Exercise 1: Creating a Simple Control Flow

► **Task 1: Set up the lab environment**

1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
2. Log on to **NY-SQL-01** as **Student** using the password **Pa\$\$w0rd**.
3. Click **Start**, and then click **My Computer**.
4. Browse to the E:\Mod03\Labfiles\Starter folder, and then double-click **Setup.cmd**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
5. Close the folder window.

► **Task 2: Create an Integration Services project**

1. Click **Start**, click **All Programs**, click **Microsoft SQL Server 2008**, and click **SQL Server Business Intelligence Development Studio**.
2. Right-click the **Start Page** tab, and then click **Close**.
3. On the **File** menu, point to **New**, and then click **Project**.
4. In the **New Project** dialog box, in the **Templates** pane, click **Integration Services Project**.
5. In the **Name** box, type **SSIS_proj3**.
6. In the **Location** box, type **E:\Mod03\Labfiles\Starter**.
7. In the **Solution Name** box, type **SSIS_sol3**.
8. Verify that the **Create directory for solution** check box is selected.
9. Click **OK**.

► **Task 3: Rename the package and the package object**

1. In Solution Explorer, right-click **Package.dtsx** (beneath the **SSIS Packages** node), and then click **Rename**.
2. In the file name box, type **Employees.dtsx**, and then press ENTER.
3. When prompted to rename the package object, click **Yes**.

► **Task 4: Create an OLE DB connection manager for the HumanResources database**

1. Double-click the **Employees.dtsx** package.
2. Right-click the **Connection Managers** pane, and then click **New OLE DB Connection**.
3. In the **Configure OLE DB Connection Manager** dialog box, click **New**.
4. In the **Connection Manager** dialog box, in the **Server name** box, type **NY-SQL-01**.
5. Verify that the **Use Windows Authentication** option is selected.
6. In the **Select or enter a database name** list, click **HumanResources**.
7. Click **Test Connection**, and then click **OK** when the connection is verified.
8. Click **OK** to close the **Connection Manager** dialog box.
9. Click **OK** to close the **Configure OLE DB Connection Manager** dialog box.
10. In the **Connection Managers** pane, right-click the new connection manager, and then click **Rename**.
11. In the connection manager name box, type **HumanResources_cm**, and then press ENTER.

- ▶ **Task 5: Create an SMTP connection manager for the local server**
 1. Right-click the **Connection Managers** pane, and then click **New Connection**.
 2. In the **Add SSIS Connection Manager** dialog box, click **SMTP**, and then click **Add**.
 3. In the **SMTP Connection Manager Editor** dialog box, in the **Name** box, type **LocalSmtp_cm**.
 4. In the **Description** field, type **Connect to local SMTP server**.
 5. In the **SMTP server** field, type **localhost**.
 6. Verify that the **Use Windows Authentication** check box and the **Enable Secure Sockets Layer (SSL)** check box are cleared, and then click **OK**.
- ▶ **Task 6: Add an Execute SQL Task to the control flow**
 1. Click the **Control Flow** tab if it is not already visible.
 2. Drag an **Execute SQL Task** from the Toolbox to the control flow design surface.

Tip: By default, the Toolbox is located on the left side of the screen. If the Toolbox is not visible, on the View menu, click Toolbox.

- 3. Double-click the task.
- 4. In the **Execute SQL Task Editor** dialog box, in the **Name** box, type **SQL Truncate Employees table**.
- 5. In the **Description** box, type **Delete data from Employees table in HumanResources database**.
- 6. In the **Connection** list, select **HumanResources_cm**.
- 7. Verify that **OLE DB** is selected in the **ConnectionType** list and **Direct input** is selected in the **SQLSourceType** list.
- 8. In the **SQLStatement** box, click the **ellipses** button.

9. In the Enter SQL Query window, type the following SQL statement:

```
TRUNCATE TABLE hr.Employees  
GO
```

10. Click **OK** to close the Enter SQL Query window.
11. In the **BypassPrepare** list, click **False**.
12. In the **Execute SQL Task Editor** dialog box, click **Parse Query**, and then click **OK** when the syntax is validated.
13. Click **OK** to close the **Execute SQL Task Editor** dialog box.
14. Click the **Save All** button on the toolbar.

► **Task 7: Add a Data Flow Task to the control flow**

1. Drag a **Data Flow Task** from the Toolbox to the control flow design surface.
2. Right-click the **Data Flow** task, and then click **Properties**.
3. In the **Properties** pane, in the **Name** field, type **DFT Retrieve employee data**.
4. In the **Description** field, type **Retrieve employee data and insert in HumanResources database**.
5. Click the **Save All** button on the toolbar.

► **Task 8: Add a Send Mail Task to the control flow**

1. Drag a **Send Mail Task** from the Toolbox to the control flow design surface.
2. Double-click the **Send Mail** task.
3. In the **Properties** pane, in the **Name** field, type **SMTP Send Successful Verification**.
4. Double-click **SMTP Send Successful Verification**.
5. In the **Description** box, type **Send e-mail message on package success**.
6. On the Mail page of the **Send Mail Task Editor** dialog box, select **LocalSmtp_cm** from the **SmtpConnection** list.
7. In the **From** box, type **sqlserver@adventure-works.com**.

8. In the **To** box, type **administrator@adventure-works.com**.
9. In the **Subject** box, type **Employee Data loaded successfully**.
10. Verify that **Direct Input** is selected in the **MessageSourceType** box.
11. In the **MessageSource** box, type **The Employees package succeeded**.
12. Verify that **Normal** is selected in the **Priority** box, and then click **OK**.
13. Click the **Save All** button on the toolbar.

► **Task 9: Copy the Send Mail task**

1. Click the **SMTP Send Successful Verification**, and on the **Edit** menu, click **Copy**.
2. Click a blank area of the design surface, and then on the **Edit** menu, click **Paste**.
3. Double-click the new **Send Mail** task.
4. On the General page of the **Send Mail Task Editor** dialog box, change the **Name** property to **SMTP Send failure notification**.
5. Change the **Description** to **Send e-mail message on package failure**.
6. On the Mail page of the **Send Mail Task Editor** dialog box, change the **Subject** to **Error loading data**, change the **MessageSource** to **The Employee package failed**, and then click **OK**.
7. Click the **Save All** button on the toolbar.

Results: After this exercise, you should have created the connection managers and tasks necessary for the Employees project.

Exercise 2: Configuring Precedence Constraints

- ▶ **Task 1: Connect a precedence constraint from the Execute SQL task to the Data Flow task**
 1. Click the Execute SQL task, and drag the precedence constraint from **SQL Truncate Employee table** to **DFT Retrieve Employee data**.
 2. Double-click the precedence constraint.
 3. Verify that **Constraint** is selected in the **Evaluation operation** list and **Success** is selected in the **Value** list, and then click **OK**.
- ▶ **Task 2: Connect precedence constraints from the Data Flow task to the Send Mail tasks**
 1. Click **DFT Retrieve Employee data**, and drag the precedence constraint from **DFT Retrieve Employee data** to the **SMTP Send success notification Send Mail** task.
 2. Double-click the precedence constraint.
 3. Verify that **Constraint** is selected in the **Evaluation operation** list and **Success** is selected in the **Value** list, and then click **OK**.
 4. Click **DFT Retrieve Employee data**, and drag a second precedence constraint from **DFT Retrieve Employee data** to the **SMTP Send failure notification Send Mail** task.
 5. Double-click the precedence constraint.
 6. Verify that **Constraint** is selected in the **Evaluation operation** list.
 7. Click **Failure** in the **Value** list, and then click **OK**.

► **Task 3: Test the control flow**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that each element except for the **SMTP Send failure notification Send Mail** task turns green.

Note: The Data Flow task contains no functionality and so completes successfully without actually doing anything.

3. On the **Debug** menu, click **Stop Debugging**.

Results: After this exercise, you should have used precedence constraints to connect the objects of the package, then been able to execute the package.

Exercise 3: Using Containers

► **Task 1: Delete the precedence constraints**

1. Right-click the precedence constraint that connects **SQL Truncate Employee table** to **DFT Retrieve Employee data**, and then click **Delete**.
2. Right-click the precedence constraint that connects **DFT Retrieve Employee data** to the **SMTP Send success notification Send Mail** task, and then click **Delete**.
3. Right-click the precedence constraint that connects **DFT Retrieve Employee data** to the **SMTP Send failure notification Send Mail** task, and then click **Delete**.

► **Task 2: Create a string variable to hold the file name**

1. Click the control flow design surface to ensure that no elements are active.
2. From the **SSIS** menu, click **Variables**.
3. In the **Variables** pane, click the **Add Variable** button.
4. In the **Name** box, in the row for the new variable, type **EmployeeFile**.

5. In the Properties pane, verify that the **Scope** for the variable is set to **Employees**.
 6. In the **Value Type** list, click **String**.
 7. Click the **Close** icon to close the **Variables** pane.
- **Task 3: Add a Sequence container to the control flow and move the Execute SQL task to that container**
1. Drag a **Sequence Container** from the Toolbox to the control flow design surface.
 2. Right-click the **Sequence** container, and then click **Properties**.
 3. In the **Name** box, in the **Properties** pane, type **SEQ Process employee data**.
 4. In the **Description** box, type **Provide container for the employee ETL operation**.
 5. Drag **SQL Truncate Employee table** from the design surface to inside the **Sequence** container.
 6. Click the **Save All** button on the toolbar.
- **Task 4: Add a Foreach Loop container to the Sequence container and move the Data Flow task into the Foreach Loop container**
1. In Solution Explorer, double-click **Employees.dtsx**.
 2. Drag a **Foreach Loop Container** from the Toolbox to inside the **Sequence** container.
 3. Double-click the **Foreach Loop** container.
 4. On the General page of the **Foreach Loop Editor** dialog box, in the **Name** field, type **FEL Enumerate employee files**.
 5. In the **Description** field, type **Implement loop to retrieve data from multiple files**.
 6. On the Collection page, click the **Enumerator** drop down and then click **Foreach File Enumerator**.
 7. In the **Folder** field, type **E:\Mod03\Labfiles\Starter\Employees**.
 8. In the **Files** field, type ***.txt**.

9. Verify that the **Fully qualified** option is selected and the **Traverse subfolders** check box is cleared.
 10. On the Variable Mappings page, in the **Variable** list, click **User::EmployeeFile**.
 11. Verify that the **Index value** is **0**, and then click **OK**.
 12. Drag **DFT Retrieve Employee data** from the design surface to inside the **ForEach Loop** container.
 13. Click the **Save All** button on the toolbar.
- **Task 5: Connect a precedence constraint from the Execute SQL task to the Foreach Loop container**
1. Click **SQL Truncate Employee table**, and drag the precedence constraint from **SQL Truncate Employee table** to the **ForEach Loop** container.
 2. Double-click the precedence constraint.
 3. Verify that **Constraint** is selected in the **Evaluation operation** list and **Success** is selected in the **Value** list, and then click **OK**.
- **Task 6: Connect precedence constraints from the Sequence container to the Send Mail tasks**
1. Click the **Sequence** container, and then drag the precedence constraint from the **Sequence** container to the **SMTP Send success notification** **Send Mail** task.
 2. Double-click the precedence constraint.
 3. Verify that **Constraint** is selected in the **Evaluation operation** list and **Success** is selected in the **Value** list, and then click **OK**.
 4. Click the **Sequence** container, and drag a second precedence constraint from the **Sequence** container to the **SMTP Send failure notification** **Send Mail** task.
 5. Double-click the precedence constraint.
 6. Verify that **Constraint** is selected in the **Evaluation operation** list.
 7. Click **Failure** in the **Value** list, and then click **OK**.

► **Task 7: Add annotations to the control flow**

1. Click the control flow design surface next to the **Sequence** container.
2. Type a description of the ETL process defined within the container.
3. Click the design surface near the **SMTP Send success notification Send Mail** task.
4. Type a description of what happens if the data extraction process succeeds.
5. Click the design surface near the **SMTP Send failure notification Send Mail** task.
6. Type a description of what happens if the data extraction process fails.
7. Click the **Save All** button on the toolbar.

► **Task 8: Run the Employees package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that each element except for the **SMTP Send failure notification Send Mail** task turns green.

Note: The Data Flow task contains no functionality and so completes successfully without actually doing anything. It is executed once for each text file in the folder searched by the Foreach Loop container.

3. On the **Debug** menu, click **Stop Debugging**.
4. Turn off virtual machine and delete changes.

Results: After this exercise, you should have used containers to implement a loop in the package which allow the package to perform the same task on multiple files.

Module 4: Implementing Data Flow

Lab: Implementing Data Flow

Exercise 1: Transferring Data

- ▶ **Task 1: Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment**
 1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
 2. Log on as **Student** with the password of **Pa\$\$w0rd**.
 3. Click **Start | Computer**, and then double-click **Allfiles (E:)**.
 4. Browse to the E:\MOD04\Labfiles\Starter folder, and then double-click **Setup.cmd**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
 5. Close the Starter window.

- ▶ **Task 2: Open the SSIS_sol4 solution**
 1. Click **Start | All Programs | Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
 2. Right-click the **Start Page** tab, and click **Close**.
 3. On the **File** menu, click **Open**, and then click **Project/Solution**.
 4. In the **Open Project** dialog box, browse to the E:\MOD04\Labfiles\Starter\SSIS_sol4 folder, and double-click the file **SSIS_sol4.sln**.
 5. Double-click **Employees.dtsx** beneath the **SSIS Packages** node in Solution Explorer to open it in the SSIS Designer.

- ▶ **Task 3: Create a connection manager for the AdventureWorks2008 database**
 1. Right-click the **Connection Managers** pane, and then click **New OLE DB Connection**.
 2. In the **Configure OLE DB Connection Manager** dialog box, click **New**.
 3. In the **Connection Manager** dialog box, verify that **Native OLE DB\SQL Native Client 10.0** is selected in the **Provider** list.
 4. In the **Server name** list, type **NY-SQL-01**.
 5. Verify that the **Use Windows Authentication** option is selected.
 6. In the **Select or enter a database name** list, click **AdventureWorks2008**.
 7. Click **Test Connection**, and then click **OK** when the connection is verified.
 8. Click **OK** to close the **Connection Manager** dialog box.
 9. Click **OK** to close the **Configure OLE DB Connection Manager** dialog box.
 10. In the **Connection Managers** pane, right-click the new connection manager, and then click **Rename**.
 11. In the **Connection manager name** box, type **AdventureWorks2008_cm**, and then press **ENTER**.

- ▶ **Task 4: Add an OLE DB Source to the DFT Create sales file data flow**
 1. On the control flow design surface, double-click the **DFT Create sales file Data Flow** task.
 2. Drag an **OLE DB Source** from the Toolbox to the data flow design surface. You may need to expand the Toolbox on the left side of the window.
 3. Right-click **OLE DB Source**, and then click **Properties**.
 4. In the **Properties** pane, in the **Name** box, type **OLE Retrieve sales data**.
 5. In the **Description** box, type **Retrieve data from SalesOrderHeader and SalesOrderDetail tables in AdventureWorks2008**.
 6. Double-click the **OLE Retrieve sales order** source.
 7. On the Connection Manager page of the **OLE DB Source Editor** dialog box, verify that **AdventureWorks2008_cm** is selected in the **OLE DB connection manager** list.

8. In the **Data access mode** list, click **SQL command**.
9. Enter the following SQL statement in the SQL command text pane:

```
SELECT h.SalesOrderID, h.SalesPersonID,
       d.OrderQty, d.ProductID, d.UnitPrice,
       d.UnitPriceDiscount, d.LineTotal
  FROM Sales.SalesOrderHeader h JOIN
       Sales.SalesOrderDetail d
      ON h.SalesOrderID = d.SalesOrderID
 WHERE d.LineTotal > 10000
```

Tip: The SalesQuery.sql file in the E:\MOD04\Labfiles\Starter\SetupFiles folder contains this SQL statement.

10. Click **Parse Query**, and then click **OK** when the syntax is validated.
11. Click **Preview**.
12. View the source data, and then click **Close**.
13. Click **OK** to close the **OLE DB Source Editor** dialog box.
14. Click the **Save All** button on the toolbar.

► **Task 5: Add a Flat File Destination to the DFT Create sales file data flow**

1. Drag a **Flat File Destination** from the Toolbox to the design surface.
2. Click the **OLE Retrieve sales data** source, and connect the green data flow path to the **Flat File Destination**.
3. Right-click the **Flat File Destination**, and then click **Properties**.
4. In the **Properties** pane, in the **Name** box, type **FFD Load sales data**.
5. In the **Description** box, type **Add data to the SalesData.csv file**.
6. Double-click the **FFD Load sales data** destination.
7. In the **Flat File Destination Editor** dialog box, click **New**.
8. In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.

9. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **SalesData_cm**.
10. In the **Description** box, type **Connects to the SalesData.csv file**.
11. In the **File** name box, type
E:\MOD04\Labfiles\Starter\Employees\SalesData.csv.
12. Select the **Column** names in the first data row check box.
13. Ensure that the following settings are selected, and then click **OK**:
 - **Locale:** English (United States)
 - **Code page:** 1252 (ANSI – Latin I)
 - **Format:** Delimited
 - **Text qualifier:** <none>
 - **Header row delimiter:** {CR}{LF}
 - **Header rows to skip:** 0
14. In the **Flat File Destination Editor** dialog box, verify the column mappings on the **Mappings** page, and then click **OK**.
15. Click the data flow design surface next to the **OLE Retrieve sales data** source.
16. Type a description of the data extraction process.
17. Click the data flow design surface next to the **FFD Load sales data** destination.
18. Type a description of the data load process.
19. Click the **Save All** button on the toolbar.

► **Task 6: Run the Employees package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that both components in the data flow turn green and that 766 rows passed through the pipeline.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 7: View the SalesData.csv file**

1. Click **Start**, and then click **Computer**.
2. In the **Computer** explorer window, navigate to the **E:\MOD04\Labfiles\Starter\Employees folder**, and verify that Integration Services created the **SalesData.csv** file.
3. Double-click the **SalesData.csv** file to open the file in Notepad. You may need to choose Notepad from a list of installed programs if prompted.
4. Verify that the file contains the sales information.
5. On the **File** menu, click **Exit**. Click **No** if prompted to save changes.
6. Minimize the Employees explorer window.

Results: After this exercise, you should have successfully added a flat file destination and verified that the SalesData.csv file contains the correct data.

Exercise 2: Implementing Transformations

► **Task 1: Create a connection manager for the HumanResources database**

1. Ensure that the **Data Flow** tab of the **Employees.dtsx** package is open in the **SSIS Designer**, then right-click the **Connection Managers** pane, and click **New OLE DB Connection**.
2. In the **Configure OLE DB Connection Manager** dialog box, click **New**.
3. In the **Connection Manager** dialog box, verify that **Native OLE DB\SQL Native Client 10.0** is selected in the **Provider** list.
4. In the **Server name** list, type **NY-SQL-01**.
5. Verify that the **Use Windows Authentication** option is selected.
6. In the **Select or enter a database name** list, click **HumanResources**.
7. Click **Test Connection**, and then click **OK** when the connection is verified.
8. Click **OK** to close the **Connection Manager** dialog box.

9. Click **OK** to close the **Configure OLE DB Connection Manager** dialog box.
 10. In the **Connection Managers** pane, right-click the new connection manager, and then click **Rename**.
 11. In the **Connection manager name** box, type **HumanResources_cm**, and then press **ENTER**.
- **Task 2: Add an OLE DB Source to the DFT Create employee files data flow**
1. On the **Data Flow** tab, click **DFT Create employee files** in the **Data Flow Task** list.
 2. Drag an **OLE DB Source** from the Toolbox to the design surface.
 3. Right-click the **OLE DB Source**, and then click **Properties**.
 4. In the **Properties** pane, in the **Name** box, type **OLE Retrieve employee data**.
 5. In the **Description** box, type **Retrieve data from Employees table in HumanResources**.
 6. Double-click the **OLE Retrieve employee data** source.
 7. On the Connection Manager page of the **OLE DB Source Editor** dialog box, in the **OLE DB connection manager** list, click **HumanResources_cm**, and then click **OK**.
 8. Verify that **Table or view** is selected in the **Data access mode** list.
 9. In the **Name of the table or the view** list, click **[hr].[Employees]**.
 10. Click **Preview**.
 11. View the source data, and then click **Close**.
 12. Click **OK** to close the **OLE DB Source Editor** dialog box.
 13. Click the data flow design surface next to the **OLE Retrieve employee data** source.
 14. Type a description of the data extraction process.
 15. Click the **Save All** button on the toolbar.

► **Task 3: Add a Lookup transformation to the DFT Create employee files data flow**

1. Drag a **Lookup** transformation from the Toolbox to the design surface.
2. Click the **OLE Retrieve employee data** source, and connect the green data flow path to the **Lookup** transformation.
3. Right-click the **Lookup** transformation, and then click **Properties**.
4. In the **Properties** pane, in the **Name** box, type **LKP Lookup e-mail addresses**.
5. In the **Description** box, type **Retrieve e-mail address from EmailAddresses table**.
6. Double-click the **LKP Lookup e-mail address** transformation.
7. On the **Connection** tab of the **Lookup Transformation Editor** dialog box, in the **OLE DB connection manager** list, click **HumanResources_cm**, and then click **OK**.
8. In the **Use a table or a view** list, click **[hr].[EmailAddresses]**.
9. Click **Preview**.
10. View the source data, and then click **Close**.
11. Click the **Columns** tab of the **Lookup Transformation Editor** dialog box, and drag a mapping between the **BusinessEntityID** column in the **Available Input Columns** list and the **BusinessEntityID** column in the **Available Lookup Columns** list.
12. In the **Available Lookup Columns** list, select the **EmailAddress** check box.
13. In the grid at the bottom of the **Columns** tab, verify that **<add as new column>** is selected in the **Lookup Operation** list for the **EmailAddress** column, and that the **Output Alias** is **EmailAddress**.
14. Click **OK** to close the **Lookup Transformation Editor** dialog box.
15. Click the data flow design surface next to the **LKP Lookup e-mail address** transformation.
16. Type a description of the lookup process.
17. Click the **Save All** button on the toolbar.

- ▶ **Task 4: Add a Sort transformation to the DFT Create employee files data flow**
 1. Drag a **Sort** transformation from the Toolbox to the design surface.
 2. Click the **LKP Lookup email addresses** transformation, and connect the green data flow path to the Sort transformation.
 3. In the **Input Output Selection** dialog box, in the **Output** list, select **Lookup Match Output**, and then **OK**.
 4. Right-click the **Sort** transformation, and then click **Properties**.
 5. In the **Properties** pane, in the **Name** box, type **SRT Sort names**.
 6. In the **Description** box, type **Sort by last name, then first name**.
 7. Double-click the **SR Sort names** transformation.
 8. In the **Sort Transformation Editor** dialog box, select the check box in front of the **LastName** column in the **Available Input Columns** list, and then select the check box in front of the **FirstName** column.
 9. In the grid on the bottom half of the **Sort Transformation Editor** dialog box, verify that **ascending** is selected in the **Sort Type** list for both columns, and then click **OK**.
 10. Click the data flow design surface next to the **SRT Sort names** transformation.
 11. Type a description of the sorting process.
 12. Click the **Save All** button on the toolbar.

- ▶ **Task 5: Add a Conditional Split transformation to the DFT Create employee files data flow**
 1. Drag a **Conditional Split** transformation from the Toolbox to the design surface.
 2. Click the **SRT Sort names** transformation, and connect the green data flow path to the **Conditional Split** transformation.
 3. Right-click the **Conditional Split** transformation, and then click **Properties**.
 4. In the **Properties** pane, in the **Name** box, type **SPL Separate employees**.
 5. In the **Description** box, type **Separate sales reps from other employees**.

6. Double-click the **SPL Separate employees** transformation.
 7. In the **Conditional Split Transformation Editor** dialog box, in the **OutputName** column of the first row in the grid, type **SalesReps**.
 8. In the **Condition** column of the first row in the grid, type **JobTitle == "Sales Representative"** (note that there are two “equals” symbols).
 9. In the **OutputName** column of the second row in the grid, type **NonSalesReps**.
 10. In the **Condition** column of the second row in the grid, type **JobTitle != "Sales Representative"**.
 11. Click **OK**.
 12. Click the data flow design surface next to the **SPL Separate employees** transformation.
 13. Type a description of the conditional split process.
 14. Click the **Save All** button on the toolbar.
- **Task 6: Add a Flat File Destination for sales reps to the DFT Create employee files data flow**
1. Drag a **Flat File** destination from the Toolbox to the design surface.
 2. Click the **SPL Separate employees** transformation, and connect the green data flow path to the **Flat File Destination**.
 3. In the **Input Output Selection** dialog box, in the **Output** list, click **SalesReps**.
 4. Verify that **Flat File Destination Input** is selected in the **Input** list, and then click **OK**.
 5. Right-click the **Flat File Destination**, and then click **Properties**.
 6. In the **Properties** pane, in the **Name** box, type **FFD Load sales reps file**.
 7. In the **Description** box, type **Add data to the SalesReps.txt file**.
 8. Double-click the **FFD Load sales reps file** destination.
 9. In the **Flat File Destination Editor** dialog box, click **New**.
 10. In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.

11. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **SalesReps_cm**.
 12. In the **Description** box, type **Connects to the SalesReps.txt file**.
 13. In the **File name** box, type
E:\MOD04\Labfiles\Starter\Employees\SalesReps.txt.
 14. Select the **Column** names in the first data row check box.
 15. Ensure that the following settings are selected, and then click **OK**:
 - **Locale:** English (United States)
 - **Code page:** 1252 (ANSI – Latin I)
 - **Format:** Delimited
 - **Text qualifier:** <none>
 - **Header row delimiter:** {CR}{LF}
 - **Header rows to skip:** 0
 16. In the **Flat File Destination Editor** dialog box, verify the column mappings on the **Mappings** page, and then click **OK**.
 17. Click the data flow design surface next to the **FFD Load sales reps file** destination.
 18. Type a description of the data loading process.
 19. Click the **Save All** button on the toolbar.
- **Task 7: Add a Flat File Destination for non-sales reps to the DFT Create employee files data flow**
1. Drag a **Flat File** destination from the Toolbox to the design surface.
 2. Click the **SPL Separate employees** transformation, and connect the unconnected green data flow path to the new **Flat File Destination**.
 3. In the **Input Output Selection** dialog box, in the **Output** list, click **NonSalesReps**.
 4. Verify that **Flat File Destination Input** is selected in the **Input** list, and then click **OK**.
 5. Right-click the **Flat File Destination**, and then click **Properties**.

6. In the **Properties** pane, in the **Name** box, type **FFD Load non-sales reps file**.
7. In the **Description** box, type **Add data to the NonSalesReps.txt file**.
8. Double-click the **FFD Load non-sales reps file** destination.
9. In the **Flat File Destination Editor** dialog box, click **New**.
10. In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.
11. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **NonSalesReps_cm**.
12. In the **Description** box, type **Connects to the NonSalesReps.txt file**.
13. In the **File** name box, type
E:\MOD04\Labfiles\Starter\Employees\NonSalesReps.txt.
14. Select the **Column** names in the first data row check box.
15. Ensure that the following settings are selected, and then click **OK**:
 - **Locale:** English (United States)
 - **Code page:** 1252 (ANSI – Latin I)
 - **Format:** Delimited
 - **Text qualifier:** <none>
 - **Header row delimiter:** {CR}{LF}
 - **Header rows to skip:** 0
16. In the **Flat File Destination Editor** dialog box, verify the column mappings on the **Mappings** page, and then click **OK**.
17. Click the data flow design surface next to the **FFD Load non-sales reps file** destination.
18. Type a description of the data loading process.
19. Click the **Save All** button on the toolbar.

► **Task 8: Run the Employees package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that the components in the data flow turn green and that 290 rows pass through the pipeline—14 sales reps and 276 other employees.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 9: View the SalesReps.txt and NonSalesReps.txt files**

1. In the Employees explorer window, verify that Integration Services created the **SalesReps.txt** and **NonSalesReps.txt** files.
2. Double-click the **SalesReps.txt** file to open the file in Notepad.
3. Verify that the file contains the correct employee information and that all records include a first and last name.
4. On the **File** menu, click **Exit**.
5. In the Employees explorer window, double-click the **NonSalesReps.txt** file to open the file in Notepad.
6. Verify that the file contains the correct employee information and that all records include a first and last name.
7. On the **File** menu, click **Exit**.
8. Minimize the Employees explorer window.

Results: After this exercise, you should have successfully created a data flow with the proper transformations and verified that the **SalesReps.txt** and **NonSalesReps.txt** files contain the correct records..

Exercise 3: Using Data Viewers

► **Task 1: Add a grid data viewer to the DFT Create sales file data flow**

1. Ensure that the **Data Flow** tab of the **Employees.dtsx** package is open in the SSIS Designer, and then in the **Data Flow Task** list, click **DFT Create sales file**.
2. Double-click the data flow path between the **OLE Retrieve sales data** and the **FFD Load sales data** destination.
3. On the Data Viewers page of the **Data Flow Path Editor** dialog box, click **Add**.
4. In the **Configure Data Viewer** dialog box, verify that **Grid** is selected in the **Type** list.
5. In the **Name** box, type **Grid: sales data**.
6. Click the **Grid** tab.
7. In the **Displayed columns** list, click **SalesPersonID**, and then click < to move the column to the **Unused columns** list.
8. Click **OK** on the **Configure Data Viewer** dialog box.

► **Task 2: Add a histogram data viewer**

1. On the Data Viewers page of the **Data Flow Path Editor** dialog box, click **Add**.
2. In the **Configure Data Viewer** dialog box, in the **Type** list, click **Histogram**.
3. In the **Name** box, type **Histogram: sales data**.
4. Click the **Histogram** tab, and then in the **Visualized column** list, click **OrderQty**.
5. Click **OK** on the **Configure Data Viewer** dialog box.

► **Task 3: Add a scatter plot data viewer**

1. On the **Data Viewers** page of the **Data Flow Path Editor** dialog box, click **Add**.
2. In the **Configure Data Viewer** dialog box, in the **Type** list, click **Scatter Plot (x,y)**.
3. In the **Name** box, type **Scatter Plot: sales data**.
4. Click the **Scatter Plot (x,y)** tab, and then in the **X axis column** list, click **OrderQty**, and in the **Y axis column** list, click **UnitPrice**.
5. Click **OK** on the **Configure Data Viewer** dialog box.

► **Task 4: Add a column chart data viewer**

1. On the **Data Viewers** page of the **Data Flow Path Editor** dialog box, click **Add**.
2. In the **Configure Data Viewer** dialog box, in the **Type** list, click **Column Chart**.
3. In the **Name** box, type **Column Chart: sales data**.
4. Click the **Column Chart** tab, and in the **Visualized column** list, click **OrderQty**.
5. Click **OK**, and then click **OK** to close the **Data Flow Path Editor** dialog box.
6. Click the **Save All** button on the toolbar.

► **Task 5: Run the package, and view the data viewers**

1. On the **Debug** menu, click **Start Debugging**.
2. View the data in the Grid: sales data at OLE Retrieve sales data. OLE DB Source Output window.

Tip: Resize the data viewers by dragging the resize handle at the bottom right corner of the window.

3. Click **Detach** to make the data flow continue while keeping the window open.
4. View the data in the Histogram: sales data at OLE Retrieve sales data. OLE DB Source Output window.
5. Click the **Close** icon to close the window.
6. View the data in the Scatter Plot: sales data at OLE Retrieve sales data. OLE DB Source Output window.
7. Click the **Close** icon to close the window.
8. View the data in the Column Chart: sales data at OLE Retrieve sales data. OLE DB Source Output window.
9. Click the **Close** icon to close the window.
10. Verify that both components in the data flow turn green and that 766 rows passed through the pipeline.
11. Click the **Close** icon in the Grid: sales data at OLE Retrieve sales data. OLE DB Source Output window.
12. On the **Debug** menu, click **Stop Debugging**

► **Task 6: Remove the data viewers**

1. Right-click the data flow path between **OLE Retrieve sales data** and **FFD Load sales data**, and then click **Data Viewers**.
2. In the table of data viewers, click **Grid: sales data**, and then click **Delete**. Repeat for the remaining data viewers.
3. Click **OK** to close the **Data Flow Path Editor** dialog box.

Results: After this exercise, you should have successfully added a data viewer of each type, viewed the data in each data viewer, and then removed the data viewers from the data flow.

Exercise 4: Configuring Error Output

► **Task 1: Add a row to the Employees table in the HumanResources database**

1. Click **Start | Computer**, and then double-click **Allfiles (E:)**.
2. Browse to the **E:\MOD04\Labfiles\Starter** folder, and then double-click **AddEmployee.cmd**. The script executes in a console window, and inserts a new record into the **hr.Employees** table with no matching record in the **hr.EmailAddresses** table. Wait for the console window to close before proceeding to the next step.
3. Close the **Starter** folder window.

► **Task 2: Add a Flat File Destination for failed rows to the DFT Create employee files data flow**

1. In the SSIS Designer, in the **Data Flow Task** list, click the **DFT Create employee files** data flow task.
2. Drag a **Flat File Destination** from the Toolbox to the design surface.
3. Click the **LKP Look up email addresses** lookup transformation, and connect the red error output path to the new **Flat File Destination**.
4. In the **Configure Error Output** dialog box, select the entire grid by clicking the **Input or Output** cell in the first row, pressing and holding SHIFT, and then clicking the **Description** cell in the last row.

5. In the **Set this value to selected cells** list, click **Redirect row**, and then click **Apply**.
6. Click **OK**.
7. Right-click the new **Flat File Destination**, and then click **Properties**.
8. In the **Properties** pane, in the **Name** box, type **FFD failed rows**.
9. In the **Description** box, type **Add failed rows to the LookupErrors.txt file**.
10. Double-click the **FFD failed rows** destination.
11. In the **Flat File Destination Editor** dialog box, click **New**.
12. In the **Flat File Format** dialog box, verify that the **Delimited** option is selected, and then click **OK**.
13. In the **Flat File Connection Manager Editor** dialog box, in the **Connection manager name** box, type **LookupErrors_cm**.
14. In the **Description** box, type **Connects to the LookupErrors.txt file**.
15. In the **File** name box, type
E:\MOD04\Labfiles\Starter\Employees\LookupErrors.txt.
16. Select the **Column** names in the first data row check box.
17. Ensure that the following settings are selected, and then click **OK**:
 - **Locale:** English (United States)
 - **Code page:** 1252 (ANSI – Latin I)
 - **Format:** Delimited
 - **Text qualifier:** <none>
 - **Header row delimiter:** {CR}{LF}
 - **Header rows to skip:** 0
18. In the **Flat File Destination Editor** dialog box, verify the column mappings on the **Mappings** page, and then click **OK**.
19. Click **Save All** on the toolbar.

► **Task 3: Run the Employees package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that the components in the data flow turn green and that 291 rows pass through the pipeline—1 error output row, 14 sales reps, and 276 other employees.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 4: View the LookupErrors.txt file**

1. In the Employees explorer window, verify that Integration Services created the **LookupErrors.txt** file.
2. Double-click the **LookupErrors.txt** file to open the file in Notepad.
3. Verify that the file contains the record for the employee with a **BusinessEntityID** value of **9999**.
4. On the **File** menu, click **Exit**.
5. Turn off the 6235A-NY-SQL-01 virtual machine and discard any changes.

Results: After this exercise, you should have successfully added a flat file destination for error output, run the package, and verified that the unmatched record appears in the **LookupError.txt** file.

Module 5: Implementing Logging

Lab: Implementing Logging

Exercise 1: Configuring Logging

- ▶ **Task 1: Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment**
 1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
 2. Log on as **Student** with the password of **Pa\$\$w0rd**.
 3. Click **Start | Computer**.
 4. In the Computer explorer window, browse to the **E:\MOD05\Labfiles\Starter** folder, and then double-click **Setup.cmd**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
 5. Close the Starter window.
- ▶ **Task 2: Open the SSIS_sol5 solution in Business Intelligence Development Studio, and then open the Employees package**
 1. Click **Start | All Programs | Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
 2. Right-click the **Start Page** tab, and then click **Close**.
 3. On the **File** menu, click **Open**, and then click **Project/Solution**.
 4. In the **Open Project** dialog box, browse to the **E:\MOD05\Labfiles\Starter\SSIS_sol5** folder, and then double-click **SSIS_sol5.sln**.
 5. Double-click **Employees.dtsx** beneath the **SSIS Packages** node in Solution Explorer to open it in SSIS Designer.

- ▶ **Task 3: Open the Configure SSIS Logs dialog box, and add the text file log provider**
 1. On the **SSIS** menu, click **Logging**.
 2. In the **Configure SSIS Logs: Employees** dialog box, in the **Provider type** list, click **SSIS log provider for Text files**, and then click **Add**.
 3. In the grid where the text provider has been added, in the **Configuration** list for that provider, click **<New connection...>**.
 4. In the **File Connection Manager Editor** dialog box, in the **Usage type** list, click **Create file**.
 5. In the **File** box, type **E:\MOD05\Labfiles\Starter\Employees\EmployeesLog.txt**, and then click **OK**.
- ▶ **Task 4: Enable logging on the DFT Create employee files Data Flow task, and enable the text file log provider for that task**
 1. In the **Containers** tree, select the **DFT Create employee files** check box. You might need to click the check box once to clear the gray, and then a second time to select it.
 2. On the **Providers and Logs** tab, select the **SSIS log provider for Text files** check box.
- ▶ **Task 5: Configure the events and information to be logged, and save the event configuration**
 1. Click the **Details** tab, and then click **Advanced**.
 2. Select the **OnError** check box, and then clear the following check boxes for that event:
 - Computer
 - Operator
 - DataBytes
 3. Repeat for the **OnInformation** and **OnWarning** events.

4. Click **Save**.
 5. In the **Save As** dialog box, navigate to the folder **E:\MOD05\Labfiles\Starter\Employees**.
 6. In the **File** name box, type **LogConfig.xml**, and then click **Save**.
- **Task 6: Enable logging on the DFT Create sales file Data Flow task, and enable the text file log provider for that task**
1. In the **Containers** tree, select the **DFT Create sales file** check box. You might need to click the check box once to clear the gray, and then a second time to select it.
 2. On the **Providers and Logs** tab, select the **SSIS log provider for Text files** check box.
- **Task 7: Use the LogConfig.xml file to configure logging on the DFT Create sales file Data Flow task**
1. Click the **Details** tab, and then click **Load**.
 2. In the **Open** dialog box, browse to the **E:\MOD05\Labfiles\Starter\Employees** folder if necessary, and then double-click **LogConfig.xml**.
 3. Click **OK** to close the **Configure SSIS Logs: Employees** dialog box.
- **Task 8: Run the Employees package, and view the contents of the EmployeesLog.txt file**
1. On the **Debug** menu, click **Start Debugging**.
 2. Verify that the control flow and data flow components turn green.
 3. On the **Debug** menu, click **Stop Debugging**.
 4. Click **Start**, and then click **Computer**.
 5. In the **Computer** explorer window, browse to the **E:\MOD05\Labfiles\Starter\Employees** folder, and then verify that Integration Services created the **EmployeesLog.txt** file.

6. Double-click **EmployeesLog.txt** to open the file in Notepad.
7. Verify that the file contains the log data. If the package ran without errors or warnings, the log file should contain data only about **OnInformation** events.
8. Close Notepad and the Employees window.

Results: After this exercise, you should have successfully configured the text file log provider, enabled logging, configured logging, run the package, and verified that the logged data appears in the EmployeesLog.txt file.

Exercise 2: Implementing Custom Logging

► **Task 1: Create a table in the HumanResources database to store the logged data**

1. Click **Start**, and then click **Command Prompt**.
2. At the command prompt, type the following command, and then press ENTER.

```
sqlcmd -i "E:\Mod05\Labfiles\Starter\SetupFiles\RowCountsTable.sql"
```

3. When the command completes, type **exit**, and then press ENTER to close the Command Prompt window.

► **Task 2: Create variables to log the number of rows inserted into each data destination**

1. In Business Intelligence Development Studio, click the **DFT Create sales file Data Flow** task on the control flow design surface, and then on the **SSIS** menu, click **Variables**.
2. In the **Variables** pane, click the **Add Variable** button.
3. In the row for the new variable, in the **Name** box, type **SalesCount**.
4. Ensure that the variable properties are configured with the following settings:
 - **Scope:** DFT Create sales file
 - **Data Type:** Int32
 - **Value:** 0

5. On the control flow design surface, click the **DFT Create employee files Data Flow** task.
 6. In the **Variables** pane, click the **Add Variable** button.
 7. In the row for the new variable, in the **Name** box, type **RepsCount**.
 8. Ensure that the variable properties are configured with the following settings:
 - **Scope:** DFT Create employee files
 - **Data Type:** Int32
 - **Value:** 0
 9. Repeat to create the **NonRepsCount** variable.
 10. Click the **Close** icon to close the **Variables** pane.
- **Task 3: Add a Row Count transformation for sales rows to the DFT Create sales file data flow**
1. Click the **Data Flow** tab.
 2. In the **Data Flow Task** list, ensure that **DFT Create sales file** is selected.
 3. Right-click the data flow path between the OLE DB source and the flat file destination, and then click **Delete**.
 4. Drag a **Row Count** transformation from the Toolbox to the data flow design surface. You may need to expand the Toolbox first.
 5. Click the **OLE Retrieve sales data** data source, and then connect the green data flow path to the **Row Count** transformation.
 6. Click the **Row Count** transformation, and then connect the green data flow path from the to the **FFD Load sales data** destination.
 7. Double-click the **Row Count** transformation.
 8. In the **Name** box, type **CNT Count sales rows**.
 9. In the **Description** box, type **Count rows and assign value to SalesCount**.
 10. In the **VariableName** box, type **SalesCount**, and then click **OK**.
 11. Click the design surface next to the **Row Count** transformation, and then type a description of the operation.
 12. Click the **Save All** button on the toolbar.

► **Task 4: Add a Row Count transformation for sales rep rows to the DFT Create employee file data flow**

1. In the **Data Flow Task** list, click **DFT Create employee files**.
2. Right-click the **SalesReps** data flow path, and then click **Delete**.
3. Drag a **Row Count** transformation from the **Toolbox** to the data flow design surface.
4. Click the **SPL Separate employees** transformation, and connect the green data flow path to the **Row Count** transformation. When prompted, select **SalesReps** in the **Output** list, and then click **OK**.
5. Click the **Row Count** transformation, and connect the green data flow path to the **FFD Load sales reps** file destination.
6. Double-click the **Row Count** transformation.
7. In the **Name** box, type **CNT Count sales reps**.
8. In the **Description** box, type **Count rows and assign value to RepsCount**.
9. In the **VariableName** box, type **RepsCount**, and then click **OK**.
10. Click the design surface next to the **Row Count** transformation, and then type a description of that operation.
11. Click the **Save All** button on the toolbar.

► **Task 5: Add a Row Count transformation for non-sales rep rows to the DFT Create sales file data flow**

1. Right-click the **NonSalesReps** data flow path, and then click **Delete**.
2. Drag a **Row Count** transformation from the **Toolbox** to the data flow design surface.
3. Click the **SPL Separate employees** transformation, and connect the unconnected green data flow path to the **Row Count** transformation. When prompted, select **NonSalesReps** in the **Output** list, and then click **OK**.
4. Click the **Row Count** transformation, and connect the green data flow path to the **FFD Load non-sales reps** file destination.
5. Double-click the **Row Count** transformation.

6. In the **Name** box, type **CNT Count non-sales reps**.
 7. In the **Description** box, type **Count rows and assign value to NonRepsCount**.
 8. In the **VariableName** box, type **NonRepsCount**, and then click **OK**.
 9. Click the design surface next to the **Row Count** transformation, and then type a description of that operation.
 10. Click the **Save All** button on the toolbar.
- **Task 6: Create an event handler for the DFT Create sales file Data Flow task**
1. Click the **Event Handlers** tab.
 2. In the **Executable** list, expand the **Employees** node, and then expand the **Executables** node.
 3. Click the **DFT Create sales file** node, and then click **OK**.
 4. In the **Event handler** list, click **OnPostExecute**.
 5. Click the hyperlink on the event handler design surface to create the event handler.
- **Task 7: Add an Execute SQL task to the event handler**
1. Drag an **Execute SQL Task** from the Toolbox to the event handler design surface.
 2. Double-click the **Execute SQL Task**.
 3. In the **Execute SQL Task Editor** dialog box, in the **Name** box, type **SQL Insert sales log data**.
 4. In the **Description** box, type **Insert log data into RowCounts table**.
 5. Under **SQL Statement**, verify that **OLE DB** is selected in the **ConnectionType** list and **Direct input** is selected in the **SQLSourceType** list, and then in the **Connection** list, click **HumanResources_cm**.

6. Click the **ellipses** button in the **SQLStatement** box, and in the Enter SQL Query window, type the following SQL statement:

```
INSERT INTO hr.RowCount
(ExecGUID, PackageName, SourceName, DestinationName, NumberOfRows)
VALUES(?, ?, ?, 'FFD Load sales data', ?)
```

7. Click **OK** to close the Enter SQL Query window.
8. Click **Parameter Mapping**, and then click **Add**.
9. In the row added to the grid, in the **Variable Name** list, click **System::ExecutionInstanceGUID**. You may need to resize the **Variable Name** column first.
10. Ensure that **Input** is selected in the **Direction** list.
11. In the **Data Type** list, click **NVARCHAR**.
12. In the **Parameter Name** list, type **0**.
13. Click **Add**, and then in the **Variable Name** list of the new row, click **System::PackageName**.
14. Ensure that **Input** is selected in the **Direction** list.
15. In the **Data Type** list, click **NVARCHAR**, and then in the **Parameter Name** list, type **1**.
16. Click **Add**, and then in the **Variable Name** list of the new row, click **System::SourceName**.
17. In the **Direction** list, ensure that **Input** is selected.
18. In the **Data Type** list, click **NVARCHAR**, and in the **Parameter Name** list, type **2**.
19. Click **Add**, and then in the **Variable Name** list of the new row, click **User::SalesCount**.
20. Ensure that **Input** is selected in the **Direction** list and **LONG** is selected in the **Data Type** list.
21. In the **Parameter Name** list, type **3**.
22. Click **OK** to close the **Execute SQL Task Editor** dialog box.
23. Click the design surface next to the **SQL Insert sales log data** task, and then type a description of the insert process.
24. Click the **Save All** button on the toolbar.

- ▶ **Task 8: Create an event handler for the DFT Create employee files Data Flow task**
 1. On the **Event Handlers** tab, click the **DFT Create employee files** node in the **Executable** list, and then click **OK**.
 2. Verify that **OnPostExecute** is selected in the **Event handler** list.
 3. Click the hyperlink on the event handler design surface to create the event handler.
- ▶ **Task 9: Add an Execute SQL task to the event handler**
 1. Click the **DFT Create sales file** node in the **Executable** list, and then click **OK**.
 2. Right-click the **SQL Insert sales log data** task, and then click **Copy**.
 3. Click the **DFT Create employee files** node in the **Executable** list, and then click **OK**.
 4. Right-click the event handler design surface, and then click **Paste**.
 5. Double-click the **SQL Insert sales log data** task.
 6. In the **Execute SQL Task Editor** dialog box, change the **Name** to **SQL Insert sales reps data**.
 7. Click the **ellipses** button in the **SQLStatement** box.
 8. In the **SQL statement**, change the value **FFD Load sales data** to **FFD Load sales reps file**, and then on the **Enter SQL Query** dialog box, click **OK**.
 9. Click **Parameter Mapping**, change the **User::SalesCount** variable to **User::RepCount**, and then click **OK**.
- ▶ **Task 10: Add a second Execute SQL task to the event handler**
 1. Right-click the **SQL Insert sales reps data** task, and then click **Copy**.
 2. Right-click the event handler design surface, and then click **Paste**.
 3. Click the **SQL Insert sales reps data** task, and connect the precedence constraint to the **SQL Insert sales reps data 1** task.
 4. Double-click the **SQL Insert sales reps data 1** task.

5. In the **Execute SQL Task Editor** dialog box, change the **Name** to **SQL Insert non-sales reps data**.
 6. Click the **ellipsis** button in the **SQLStatement** box.
 7. In the SQL statement, change the value **FFD Load sales reps file** to **FFD Load non-sales reps file**, and then on the **Enter SQL Query** dialog box, click **OK**.
 8. Click **Parameter Mapping**, change the **User::RepCount** variable to **User::NonRepCount**, and then click **OK**.
 9. Click the design surface, and then type a description of the insert process.
 10. Click the **Save All** button on the toolbar.
- **Task 11: Run the Employees package, and view the contents of the RowCounts table**
1. On the **Debug** menu, click **Start Debugging**.
 2. Verify that the control flow and data flow components turn green.
 3. On the **Debug** menu, click **Stop Debugging**.
 4. Click **Start | All Programs | Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
 5. In the **Connect to Server** dialog box, ensure that **Database Engine** is selected in the **Server type** list, verify that **NY-SQL-01** is selected in the **Server name** box, verify that **Windows Authentication** is selected in the **Authentication** box, and then click **Connect**.
 6. In Object Explorer, expand **Databases | HumanResources | Tables**.
 7. Right-click **hr.RowCount**, and then click **Select Top 1000 Rows**.
 8. View the table contents and verify that the row counts have been logged.
 9. Close SQL Server Management Studio, and then shutdown the virtual machine without saving changes to reset it for the next Lab.
 10. Turn off the 6235A-NY-SQL-01 virtual machine and discard any changes.

Results: After this exercise, you should have successfully added variables, added Row Count destinations, created event handlers, added Execute SQL tasks, run the package, and verified that the row counts appear in the database table.

Module 6: Debugging and Error Handling

Lab: Debugging and Error Handling

Exercise 1: Debugging a Package

- ▶ **Task 1: Start the 6235A-NY-SQL-01 virtual machine, log on as Student, and set up the lab environment**
 1. In the Lab Launcher, next to **6235A-NY-SQL-01**, click **Launch**.
 2. Log on as **Student** with the password of **Pa\$\$w0rd**.
 3. Click **Start | Computer**.
 4. In the **Computer** explorer windows, browse to the **E:\MOD06\Labfiles\Starter** folder, and then double-click **Setup.cmd**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
 5. Close the Starter window.
- ▶ **Task 2: Open the SSIS_sol6 solution in Business Intelligence Development Studio**
 1. Click **Start | All Programs | Microsoft SQL Server 2008**, right-click **SQL Server Business Intelligence Development Studio**, and then click **Run As Administrator**. Enter **Pa\$\$w0rd** as the password.
 2. Right-click the **Start Page** tab, and then click **Close**.
 3. On the **File** menu, click **Open**, and then click **Project/Solution**.
 4. In the **Open Project** dialog box, browse to the **E:\MOD06\Labfiles\Starter\SSIS_sol6** folder, and then double-click **SSIS_sol6.sln**.
 5. In Solution Explorer, in the **SSIS Packages** folder, double-click **Employees.dtsx**.

► **Task 3: Add a breakpoint to the Data Flow task**

1. Right-click the **DFT Retrieve employee data** task, and then click **Edit Breakpoints**.
2. In the **Set Breakpoints – DFT Retrieve employee data** dialog box, select the check box for the **Break when the container receives the OnPreExecute event break condition**.
3. Verify that **Always** is selected in the **Hit Count Type** list for the selected break condition.
4. Click **OK**. The **Data Flow** task now displays a red dot.

► **Task 4: Add a breakpoint to the script**

1. Double-click the **SCR Add timestamp** task.
2. On the **Script** page of the **Script Task Editor** dialog box, click **Edit Script**.
3. In the Integration Services Script Task window, in the Project Explorer pane, double-click **ScriptMain.vb**.
4. Click the gray margin to the left of the following line of code: (You may need to scroll down.)

```
Dim sw As StreamWriter
```

Verify that the margin now contains a solid red circle to the left to the line of code, and that part of the code is highlighted in red.

5. Close the Integration Services Script Task window.
6. Click **OK** to close the **Script Task Editor** dialog box. The **Script** task now displays a red dot.

► **Task 5: Add a data viewer to the package**

1. Double-click the **DFT Retrieve employee data** task.
2. On the data flow design surface, double-click the data flow path that connects the **FFS Extract employee data** Flat File source to the **CNV Convert data types** Data Conversion transformation.
3. On the Data Viewers page of the **Data Flow Path Editor** dialog box, click **Add**.
4. In the **Configure Data Viewer** dialog box, in the **Type** list, verify that **Grid** is selected.
5. In the **Name** box, type **Grid data viewer**, and then click **OK**.
6. Click **OK** to close the **Data Flow Path Editor** dialog box.

► **Task 6: Enable Integration Services logging, and display the Log Events pane**

1. On the **SSIS** menu, click **Logging**.
2. In the **Configure SSIS Logs: Employees** dialog box, in the **Containers** list, select the **Employees** check box, and then click the **Details** tab.
3. In the **Events** list, select the **OnError** check box and the **OnPostExecute** check box, and then click **OK**.
4. On the **SSIS** menu, click **Log Events** to open the **Log Events** pane.

► **Task 7: Run the package and monitor events, row counts, data samples, and variable values as they change**

1. Click the **Control Flow** tab.
2. On the **Debug** menu, click **Start Debugging**. When the package starts to run, SSIS Designer opens the debugging windows, the Microsoft Visual Studio for Applications window, and the data viewer window, which contains no data at this point.
3. Wait for the package to stop at the first break point.
4. In the debugging windows, click the **Locals** tab, and then expand the **Variables** node.
5. Right-click the **User::EmployeeFile** variable, and then click **Add Watch**.

6. Verify that the value of the **EmployeeFile** variable in the Watch 1 debugging window is {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\Employees001-100.txt}.
7. On the **Debug** menu, click **Continue**. The package stops running to display data in the data viewer window. You might need to switch back from the Microsoft Visual Studio for Applications window to SSIS Designer to display the data viewer window.
8. Verify that the data viewer window contains the first 100 rows of employee data. Then click the **Data Flow** tab to view the number of rows that moved through the data flow. There should be 100 rows in the data flow.
9. In the data viewer window, click the **Continue** button. SSIS Designer now displays the Microsoft Visual Studio for Applications window and stops running the package at the breakpoint. The line of code next to the breakpoint is now highlighted in yellow, and the red dot contains a yellow arrow.
10. On the **Debug** menu in Visual Studio for Applications, click **Windows**, and then click **Locals**.
11. In the Locals debugging window, verify that the value of the **EmployeeFile** variable is {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\Employees001-100.txt}.
12. On the **Debug** menu in Visual Studio for Applications, click **Continue**. SSIS Designer stops at the breakpoint on the **Data Flow** task.
13. Verify that the value of the **EmployeeFile** variable in the Watch 1 debugging window is now {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\Employees101-200.txt}. You may need to move the data viewer window out of the way.
14. On the **Debug** menu, click **Continue**. The package stops running to display data in the data viewer window. Again, you might need to switch back from the Microsoft Visual Studio for Applications window to SSIS Designer to display the data viewer window.
15. Verify that the data viewer window now displays the next set of rows of employee data, and then click the **Continue** button in the data viewer window. SSIS Designer again stops at the breakpoint in the script.
16. Verify that the value of the **EmployeeFile** variable in the Locals debugging window is now {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\Employees101-200.txt}.

17. On the **Debug** menu in Visual Studio for Applications, click **Continue**. SSIS Designer stops at the breakpoint on the **Data Flow** task.
18. Verify that the value of the **EmployeeFile** variable in the Watch 1 debugging window is now {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\ Empl Employees201-300.txt}.
19. On the **Debug** menu, click **Continue**. The package stops running to display data in the data viewer window. Again, you might need to switch back from the Microsoft Visual Studio for Applications window to SSIS Designer to display the data viewer window.
20. Verify that the data viewer window now displays the next set of rows of employee data. There are only 90 records in the third employee file.
21. Click the **Data Flow** tab to view the number of rows that moved through the data flow. Again, you should see 90 rows.
22. In the data viewer window, click the **Close** icon. SSIS Designer stops at the breakpoint in the script.
23. Verify that the value of the **EmployeeFile** variable in the Locals debugging window is now {E:\\MOD06\\Labfiles\\Starter\\SetupFiles\\ Empl Employees201-300.txt}.
24. On the **Debug** menu, click **Stop Debugging**.
25. In the **Log Events** pane, review the events logged during package execution.

► **Task 8: Remove the breakpoints and the data viewer**

1. On the **Data Flow** tab, right-click the data flow path that contains the data viewer, and then click **Data Viewers**.
2. On the **Data Viewers** tab of the **Data Flow Path Editor** dialog box, in the **Data viewers** list, click **Grid data viewer**.
3. Click **Delete**, and then click **OK**.
4. Click the **Control Flow** tab.
5. On the **Debug** menu, click **Delete All Breakpoints**, and then click **Yes** when prompted to confirm the deletion.

Results: After this exercise, you should have successfully configured the breakpoints, enabled logging, configured logging, run the package, and verified that the data is viewable at each breakpoint.

Exercise 2: Implementing Error Handling

► **Task 1: Create a table in the HumanResources database to log errors**

1. Click **Start**, and then click **Command Prompt**.
2. At the command prompt, type the following command, and then press ENTER:

```
sqlcmd -i "E:\MOD06\Labfiles\Starter\SetupFiles\LogEventsTable.sql"
```
3. When the command completes, type **exit**, and then press ENTER to close the Command Prompt window.

- ▶ **Task 2: Add an event handler to capture errors generated by the Foreach Loop container and its child tasks**
 1. Click the **Event Handlers** tab.
 2. In the **Executable** list, expand the **Employees** node, and then expand the **Executables** node.
 3. Click the **FEL Enumerate employee files** node, and then click **OK**.
 4. Verify that **OnError** is selected in the **Event handler** list.
 5. Click the hyperlink on the event handler design surface to create the event handler.
- ▶ **Task 3: Configure the event handler to prevent propagation**
 1. On the **SSIS** menu, click **Variables**.
 2. In the **Variables** pane, click the **Show System Variables** and **Show All Variables** buttons.
 3. In the row that contains the **Propagate** variable, in the **Value** list, click **False**. You may need to resize the **Variables** pane first.
 4. Click the **Close** icon to close the **Variables** pane.
- ▶ **Task 4: Add an Execute SQL Task to the event handler**
 1. Drag an **Execute SQL Task** from the **Toolbox** to the event handler design surface.
 2. Double-click the task.
 3. On the General page of the **Execute SQL Task Editor** dialog box, in the **Name** box, type **SQL Log errors**.
 4. In the **Description** box, type **Log error data and file name**.
 5. Verify that **OLE DB** is selected in the **ConnectionType** list and **Direct input** is selected in the **SQLSourceType** list.
 6. In the **Connection** list, click **HumanResources_cm**.
 7. In the **SQLStatement** box, click the **Ellipses** button.

8. In the Enter SQL Query window, type the following SQL statement:

```
INSERT INTO hr.LogEvents
    (ExecGUID, PackageName, SourceName, ErrorCode, ErrorDescription,
EmployeeFile)
VALUES (?, ?, ?, ?, ?, ?)
```

9. Click **OK** to close the **Enter SQL Query** window.
10. On the **Parameter Mapping** page, click **Add**.
11. In the row added to the grid, in the **Variable Name** list, click **System::ExecutionInstanceGUID**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, click **NVARCHAR**, and then in the **Parameter Name** list, type **0**.
12. Click **Add**, and then in the **Variable Name** list of the new row, click **System::PackageName**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, click **NVARCHAR**, and in the **Parameter Name** list, type **1**.
13. Click **Add**, and then in the **Variable Name** list of the new row, click **System::SourceName**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, click **NVARCHAR**, and in the **Parameter Name** list, type **2**.
14. Click **Add**, and then in the **Variable Name** list of the new row, click **System::ErrorCode**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, ensure that **LONG** is selected, and in the **Parameter Name** list, type **3**.
15. Click **Add**, and then in the **Variable Name** list of the new row, click **System::ErrorDescription**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, click **NVARCHAR**, and in the **Parameter Name** list, type **4**.
16. Click **Add**, and then in the **Variable Name** list of the new row, click **User::EmployeeFile**. In the **Direction** list, ensure that **Input** is selected, in the **Data Type** list, click **NVARCHAR**, and in the **Parameter Name** list, type **5**.
17. Click **OK** to close the **Execute SQL Task Editor** dialog box.
18. Click the design surface next to the **SQL Log errors** task, and then type a description of the insert process.
19. Click the **Save All** button on the toolbar.

► **Task 5: Add a Try-Catch block to the script in the Script task**

1. Click the **Control Flow** tab, and then double-click the **SCR Add timestamp** task.
2. On the **Script** page of the **Script Task Editor** dialog box, click **Edit Script**.
3. In the Integration Services Script Task window, in Project Explorer, double-click **ScriptMain.vb**.
4. Modify the **Main()** function in the script to include a **Try-Catch** block, as shown in the following code:

```
Public Sub Main()
    'Create a Try/Catch block
    Try
        'Define variable to hold file name
        Dim EmployeeFile As String = _
            DirectCast(Dts.Variables("EmployeeFile").Value,
String)
        'Define variable to hold the execution start time
        Dim StartTime As String = _
            CType(Dts.Variables("StartTime").Value, String)
        'Define a StreamWriter variable to write to the file
        Dim sw As StreamWriter
        'Use the AppendText method to write to the employee
        file
        sw = File.AppendText(EmployeeFile)
        'Use the WriteLine method to add the text
        sw.WriteLine("")
        sw.WriteLine("*** DATA LOADED " + StartTime + " ***")
        'Flush and close the StreamWriter object
        sw.Flush()
        sw.Close()
    Catch ex As Exception
        'Raise an error event if exception
        Dts.Events.FireError(-1, "Script task",
            "Cannot write to employee file. " + ex.Message, "")
    End Try
    'Inform runtime that task succeeded
    Dts.TaskResult = ScriptResults.Success
End Sub
```

5. Close the Integration Services Script Task window.
6. Click **OK** to close the **Script Task Editor** dialog box.
7. Click the **Save All** button on the toolbar.

► **Task 6: Modify Integration Services logging to log only errors**

1. On the **SSIS** menu, click **Logging**.
2. In the **Configure SSIS Logs: Employees** dialog box, click the **Details** tab.
3. Clear the **OnPreExecute** check box in the **Events** list, and then click **OK**.

► **Task 7: Configure the employee text files as read-only**

1. Click **Start**, and then click **Computer**.
2. Browse to the **E:\MOD06\Labfiles\Starter\SetupFiles** folder.
3. Click the **Employees001-100.txt** file, press SHIFT, and then click the **Employees201-300.txt** file.
4. Right-click the files, and then click **Properties**.
5. Select the **Read-only** check box, and then click **OK**.

► **Task 8: Run the package, and view the events in the Log Events pane**

1. In Business Intelligence Development Studio, click the **Control Flow** tab if it is not selected.
2. On the **Debug** menu, click **Start Debugging**.
3. In the Integration Service Script Task window, on the **Debug** menu, click **Continue**.
4. In Business Intelligence Development Studio, on the **Debug** menu, click **Stop Debugging**.
5. View the information about the **OnError** events in the Log Events pane.

► **Task 9: View the data in the Employees and LogEvents tables**

1. Click Start | All Programs | Microsoft SQL Server 2008, and then click **SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, ensure that **Database Engine** is selected in the **Server type** list, **NY-SQL-01** is selected in the **Server name** box, and **Windows Authentication** is selected in the **Authentication** box. Then click **Connect**.
3. In Object Explorer, expand **Databases**, **HumanResources**, and **Tables**.
4. Right-click the **hr.Employees** table, and then click **Select Top 1000 Rows**. It should contain 100 rows, because the package failed when an error occurred after the first file had been loaded.
5. Right-click the **hr.LogEvents** table, and then click **Select Top 1000 Rows**. It should contain one row for the error logged by the event handler. (Note that if you ran the package multiple times during the lab, you may see more than one row.)
6. Keep the tables open and minimize Microsoft SQL Server Management Studio.

Results: After this exercise, you should have successfully added a try/catch block, configured the variable to log, run the package, and then confirmed the entries in the **hr.LogEvents** table.

Exercise 3: Controlling Failure Behavior

- ▶ **Task 1: Configure the MaximumErrorCount property of the Foreach Loop container**
 1. Right-click the **Foreach Loop** container, and then click **Properties**.
 2. Type **3** as the value for the **MaximumErrorCount** property.
- ▶ **Task 2: Run the package, and view the events in the Log Events pane**
 1. On the **Debug** menu, click **Start Debugging**.
 2. In the Integration Services Script Task window, on the **Debug** menu, click **Continue**.
 3. In Business Intelligence Development Studio, on the **Debug** menu, click **Stop Debugging**.
 4. View the information about the **OnError** events in the **Log Events** pane.
- ▶ **Task 3: View the data in the Employees and LogEvents tables**
 1. In SQL Server Management Studio, click the **Table-hr.Employees** tab, and then click the **Execute SQL** button. The table should contain 290 rows, because all three files were loaded.
 2. Click the **Table-hr.LogEvents** tab, and then click the **Execute SQL** button. The table should contain 3 new rows, one for each error logged by the event handler.
 3. Keep the tables open and minimize SQL Server Management Studio.

► **Task 4: Configure the employee text files as read-write**

1. In Windows Explorer, navigate to the E:\MOD06\Labfiles\Starter\SetupFiles folder.
2. Click the **Employees001-100.txt** file, press SHIFT, and then click the **Employees201-300.txt** file.
3. Right-click the files, and then click **Properties**.
4. Clear the **Read-only** check box, and then click **OK**.
5. Close Windows Explorer.

► **Task 5: Configure the ForceExecutionResult property of the Script task**

1. In Business Intelligence Development Studio, click the **Control Flow** tab if it is not selected.
2. Right-click the **Script** task, and then click **Properties**.
3. Set the **ForceExecutionResult** property to **Failure**.

► **Task 6: Run the package, and view the events in the Log Events pane**

1. On the **Debug** menu, click **Start Debugging**.
2. In the Integration Services Script Task window, on the **Debug** menu, click **Continue**.
3. In Business Intelligence Development Studio, on the **Debug** menu, click **Stop Debugging**.
4. View the events in the **Log Events** pane.

► **Task 7: View the data in the Employees and LogEvents tables**

1. In SQL Server Management Studio, click the **Table-hr.Employees** tab, and then click the **Execute SQL** button. The table should contain 290 rows, because all three files were loaded.
2. Click the **Table-hr.LogEvents** tab, and then click the **Execute SQL** button. The table should only contain two new rows, even though the script task was forced to fail.
3. Close SQL Server Management Studio and Business Intelligence Development Studio.
4. Turn off the 6235A-NY-SQL-01 virtual machine and discard any changes

Results: After this exercise, you should have successfully modified the package failure behavior, run the package, and verified that the correct rows appear in the database tables.

Module 7: Implementing Checkpoints and Transactions

Lab: Implementing Checkpoints and Transactions

Exercise 1: Implementing Checkpoints in a Package

► **Task 1: Configure the lab environment**

1. In the Lab Launcher, next to **6235A-NY-SQL-01**, click **Launch**.
2. Log on to **NY-SQL-01** as **Student** using the password **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. Browse to the **E:\MOD07\Labfiles\Starter** folder, and then double-click the **Setup.cmd** file.
5. Notice that the script executes in a console window. Ignore any error messages displayed in the console window. Wait for the console window to close before proceeding to the next step.
6. On the **Start** menu, point to **Administrative Tools** and then click **Services**.
7. The **User Account Control** dialog box appears. Click **Continue**.
8. The Services window opens. In the right pane, click **Simple Mail Transfer Protocol (SMTP)**, and then click the **Start** button.
9. Close Services.
10. Close the Windows Explorer window.

► **Task 2: View the database data**

1. Click **Start | All Programs | Microsoft SQL Server 2008 | SQL Server Management Studio**.
2. The Microsoft SQL Server Management Studio window opens. The **Connect to Server** dialog box appears. Click **Connect**.
3. In Object Explorer, expand **Databases | HumanResources | Tables**.

4. Right-click **hr.Employees1**, and then click **Select Top 1000 Rows**.
5. View the contents of the table. It should be empty.
6. Keep the table open and minimize Microsoft SQL Server Management Studio.

► **Task 3: Open an SSIS solution**

1. Click **Start**, click **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. Right-click the **Start Page** tab, and then click **Close**.
3. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. The **Open Project** dialog box appears. Browse to the **E:\MOD07\Labfiles\Starter\SSIS_sol7** folder.
5. Click **SSIS_sol7.sln**, and then click **Open**.
6. In Solution Explorer, double-click **Employees1.dtsx** beneath the **SSIS Packages** node.
7. Examine the structure and comments on the control flow surface in order to understand what the package does.

► **Task 4: Configure the checkpoint properties on the package**

1. Right-click the control flow design surface, and then click **Properties**.
2. In the **Properties** pane, in the **CheckpointFileName** field, type **E:\MOD07\Labfiles\Starter\Employees\EmployeeCheckpoint.txt**.
3. In the **CheckpointUsage** list, click **IfExists**.
4. In the **SaveCheckpoints** list, click **True**.

► **Task 5: Configure the FailPackageonFailure property on the tasks in the Sequence container**

1. In the control flow design surface, in the **SEQ Process employee data Sequence** container, right-click the **SQL Truncate Employees1 Execute** task, and then click **Properties**.
2. In the **Properties** pane, in the **FailPackageOnFailure** field, click **True**.
3. In the control flow design surface, in the **SEQ Process employee data Sequence** container, right-click the **DFT Retrieve employee data Data Flow** task, and then click **Properties**.
4. In the **Properties** pane, in the **FailPackageOnFailure** field, click **True**.
5. In the control flow design surface, in the **SEQ Process employee data Sequence** container, right-click the **SCR Add timestamp Script** task, and then click **Properties**.
6. In the **Properties** pane, in the **FailPackageOnFailure** field, click **True**.
7. On the toolbar, click the **Save All** button.

► **Task 6: Run the package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that the **SQL Truncate Employees1 Execute** SQL task and **DFT Retrieve employee data Data Flow** task turn green, and that the **SEQ Process employee data Sequence** container and **SCR Add timestamp Script** task turn red.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 7: View the database data**

1. In the Microsoft SQL Server Management Studio, on the toolbar, click **Execute**.
2. View the contents of the table. It should contain 290 rows.
3. Keep the table open and minimize SQL Server Management Studio.

► **Task 8: View the output file**

1. Click **Start**, and then click **Computer**.
2. In Microsoft Windows Explorer, navigate to the **E:\MOD07\Labfiles\Starter\Employees** folder, and verify that Integration Services created the **EmployeeCheckpoint.txt** file.
3. Double-click the **EmployeeCheckpoint.txt** file to open the file in Notepad.
4. The Notepad window opens. Verify that the file contains checkpoint data.
5. Close Notepad.

► **Task 9: Configure the output file as read-write**

1. In Windows Explorer, right-click the **Employees1.txt** file, and then click **Properties**.
2. The **Employees1.txt Properties** dialog box appears. Clear the **Read-only** check box, and then click **OK**.

► **Task 10: Re-run the package**

1. In Business Intelligence Development Studio, on the **Debug** menu, click **Start Debugging**.
2. Verify that the **SQL Truncate Employees1** Execute SQL task and **DFT Retrieve employee data** Data flow task do not change color and that the **SEQ Process employee data** Sequence container, **SCR Add timestamp** Script task, and **SEM Send e-mail on success** Send Mail task turn green.
3. On the **Debug** menu, click **Stop Debugging**.
4. Keep Business Intelligence Development Studio open. You will use it in the next exercise.

► **Task 11: View database data and the output file**

1. In SQL Server Management Studio, on the toolbar, click **Execute**.
2. View the contents of the table. The table should still contain only 290 rows. If the checkpoints had not provided the correct starting point for the package execution, that table might include no rows or double the number of rows (580 rows).
3. Keep the SQL Server Management Studio open. You will use it in the next exercise.
4. In Windows Explorer, navigate to the **E:\MOD07\Labfiles\Starter\Employees** folder, and then double-click the **Employees1.txt** file to open the file in Notepad.
5. The Notepad window opens. Scroll to the bottom of the document, and verify that a timestamp has been added.
6. Close Notepad.

► **Task 12: Verify that an email message was sent**

1. In Microsoft Windows Explorer, navigate to the **C:\inetpub\mailroot\Drop** folder.
2. If that folder is empty, navigate to the **C:\inetpub\mailroot\Queue** folder.
3. In the details pane, double-click the **Internet E-Mail Message** file.
4. The **Windows** dialog box appears. Click **Select a program from a list of installed programs**, and then click **OK**.
5. The **Open With** dialog box appears. Click **Notepad**, and then click **OK**.
6. The Notepad window opens.
7. Verify that the **SMT Send e-mail on success Send Mail** task has sent this e-mail message. Notice the **From**, **To**, and **Subject** headers.
8. Close the Notepad window.
9. Keep Windows Explorer open. You will use it in the next exercise.

Results: After this exercise, the checkpoints have been implemented in the package.

Exercise 2: Implementing Transactions in a Package

► Task 1: View the database data

1. In SQL Server Management Studio, in Object Explorer, expand **Databases** | **HumanResources** | **Tables** if necessary.
2. Right-click **hr.Employees2**, and then click **Select Top 1000 Rows**.
3. View the contents of the table. It should contain 10 rows.
4. Keep the table open and minimize SQL Server Management Studio.

► Task 2: Configure the Sequence container to start a transaction

1. In Business Intelligence Development Studio, in Solution Explorer, double-click **Employees2.dtsx** beneath the **SSIS Packages** node.
2. Read the comments on the control flow surface to understand what the package does.
3. On the control flow design surface, right-click the **SEQ Process employee data** **Sequence** container, and then click **Properties**.
4. In the **Properties** pane, in the **TransactionOption** list, click **Required**.

► Task 3: Configure the ValidateExternalMetadata property on the OLE DB destination

1. On the control flow design surface, double-click the **DFT Retrieve employee data** **Data Flow** task.
2. Right-click the **OLE Load employee data** OLE DB destination, and then click **Properties**.
3. In the **Properties** pane, in the **ValidateExternalMetadata** list, click **False**.
4. On the toolbar, click the **Save All** button.

► **Task 4: Run the package**

1. Click the **Control Flow** tab for the **Employees2.dtsx** package.
2. On the **Debug** menu, click **Start Debugging**.
3. Verify that the **SQL Truncate Employees2 Execute SQL** task and **DFT Retrieve employee data Data flow** task turn green and that the **SEQ Process employee data Sequence** container and **SCR Add timestamp Script** task turn red.
4. On the **Debug** menu, click **Stop Debugging**.

► **Task 5: View the database data**

1. In Microsoft SQL Server Management Studio, in Object Explorer, expand **Databases** | **HumanResources** | **Tables** if necessary.
2. Right-click **hr.Employees2**, and then click **Select Top 1000 Rows**.
3. View the contents of the table. It should still contain 10 rows. If the Integrations Services had not properly rolled back the transaction, the table would contain 290 rows.
4. Keep the table open and minimize SQL Server Management Studio.

► **Task 6: Configure the output file as read-write**

1. In Microsoft Windows Explorer, navigate to the **E:\MOD07\Labfiles\Starter\Employees** folder.
2. Right-click the **Employees2.txt** file, and then click **Properties**.
3. The **Employees2.txt Properties** dialog box appears. Clear the **Read-only** check box, and then click **OK**.

► **Task 7: Re-run the package**

1. In Business Intelligence Development Studio, on the **Debug** menu, click **Start Debugging**.
2. Verify that all tasks and the container turn green.
3. On the **Debug** menu, click **Stop Debugging**.
4. Keep Business Intelligence Development Studio open. You will use it in the next exercise.

► **Task 8: View the database data and the output file**

1. In SQL Server Management Studio, on the toolbar, click **Execute**.
2. View the content of the table. The table should now contain 290 rows.
3. Keep Microsoft SQL Server Management Studio open. You will use it in the next exercise.
4. In Windows Explorer, double-click the **Employees2.txt** file to open the file in Notepad.
5. The Notepad window opens. Scroll to the bottom of the document, and verify that a timestamp has been added.
6. Close Notepad.
7. Keep Windows Explorer open. You will use it in the next exercise.

► **Task 9: Verify that an email message was sent**

1. In Microsoft Windows Explorer, navigate to the **C:\inetpub\mailroot\Drop** folder.
2. If that folder is empty, navigate to the **C:\inetpub\mailroot\Queue** folder.
3. In the details pane, double-click the most recent **Internet E-Mail Message** file.
4. The Notepad window opens.
5. Verify that the **SMT Send e-mail on success Send Mail** task has sent this e-mail message. Notice the **From**, **To**, and **Subject** headers.

6. Close the Notepad window.
7. Keep Windows Explorer open. You will use it in the next exercise.

Results: After this exercise, the transactions have been implemented in the package.

Exercise 3: Implementing a Native Transaction

► **Task 1: View the database data**

1. In Microsoft SQL Server Management Studio, in Object Explorer, expand **Databases | HumanResources | Tables** if necessary.
2. Right-click **hr.Employees3**, and then click **Select Top 1000 Rows**.
3. View the contents of the table. It should contain 10 rows.
4. Keep the table open and minimize SQL Server Management Studio.

► **Task 2: Add an Execute SQL Task to the Sequence container in the package**

1. In Business Intelligence Development Studio, in Solution Explorer, double-click **Employees3.dtsx** beneath the **SSIS Packages** node.
2. Read the comments on the control flow surface to understand what the package does.
3. Drag an **Execute SQL Task** from the Toolbox and drop it inside the **SEQ Process employee data Sequence** container above the existing tasks.
4. Connect the green precedence constraint from the new task to the **SQL Truncate Employees3 Execute SQL** task.
5. Double-click the new **Execute SQL** task.
6. The **Execute SQL Task Editor** dialog box appears. On the General page, in the **Name** field, type **SQL Begin Transaction**.
7. In the **Description** field, type **Start transaction at beginning of container**.
8. In the **Connection** list, select **HumanResources_cm**.
9. Verify that **OLE DB** is selected in the **ConnectionType** list and that **Direct input** is selected in the **SQLSourceType** list.
10. In the **SQLStatement** field, click the **ellipses** button.

11. The **Enter SQL Query** dialog box appears. Type the following SQL statement in the **Enter SQL Query** window:

```
BEGIN TRANSACTION EmpTran  
GO
```

12. Click **OK** to close the Enter SQL Query window.
13. Click **OK** to close the **Execute SQL Task Editor** dialog box.

► **Task 3: Add another Execute SQL Task to the Sequence container**

1. Drag an **Execute SQL Task** from the Toolbox to inside the **Sequence** container and drop it inside the **SEQ Process employee data Sequence** container below the existing tasks.
2. Click the **SCR Add timestamp Script** task, and connect the green precedence constraint to the new **Execute SQL** task.
3. Double-click the new **Execute SQL** task.
4. The **Execute SQL Task Editor** dialog box appears. On the **General** page, in the **Name** field, type **SQL Commit transaction**.
5. In the **Description** field, type **Commit transaction at end of container**.
6. Verify that **OLE DB** is selected in the **ConnectionType** list and that **Direct input** is selected in the **SQLSourceType** list.
7. In the **Connection** list, select **HumanResources_cm**.
8. In the **SQLStatement** field, click the **ellipses** button.
9. The **Enter SQL Query** dialog box appears. Type the following SQL statement in the **Enter SQL Query** window:

```
COMMIT TRANSACTION EmpTran  
GO
```

10. Click **OK** to close the Enter SQL Query window.
11. Click **OK** to close the **Execute SQL Task Editor** dialog box.

► **Task 4: Configure the RetainSameConnection property on the connection manager**

1. In the **Connection Managers** pane, right-click the **HumanResources_cm** connection manager, and then click **Properties**.
2. In the **Properties** pane, in the **RetainSameConnection** list, click **True**.
3. On the toolbar, click the **Save All** button.

► **Task 5: Run the package**

1. On the **Debug** menu, click **Start Debugging**.
2. Verify that the first three tasks in the **Sequence** container turn green and that the **SCR Add timestamp** Script task and the **Sequence** container turn red.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 6: View the database data**

1. In the Microsoft SQL Server Management Studio, on the toolbar, click **Execute**.
2. Examine the contents of the table. The table should still contain only 10 rows. If SQL Server had not properly rolled back the transaction, the table would contain 290 rows.

► **Task 7: Configure the output file as read-write**

1. In Microsoft Windows Explorer, navigate to the **E:\MOD07\Labfiles\Starter\Employees** folder.
2. Right-click the **Employees3.txt** file, and then click **Properties**.
3. The **Employees3.txt Properties** dialog box appears. Clear the **Read-only** check box, and then click **OK**.

► **Task 8: Re-run the package**

1. In Business Intelligence Development Studio, on the **Debug** menu, click **Start Debugging**.
2. Verify that all tasks and the container turn green.
3. On the **Debug** menu, click **Stop Debugging**.

► **Task 9: View database data and the output file**

1. In the Microsoft SQL Server Management Studio, on the toolbar, click **Execute**.
2. Examine the query results. The table should now contain 290 rows.
3. In Windows Explorer, double-click the **Employees3.txt** file to open the file in Notepad.
4. The Notepad window opens. Scroll to the bottom of the document, and verify that the timestamp has been added.
5. Close Notepad.

► **Task 10: Verify that an email message was sent**

1. In Microsoft Windows Explorer, navigate to the **C:\inetpub\mailroot\Drop** folder.
2. If that folder is empty, navigate to the **C:\inetpub\mailroot\Queue** folder.
3. In the details pane, double-click the most recent **Internet E-Mail Message** file.
4. The Notepad window opens.
5. Verify that the **SMT Send e-mail on success Send Mail** task has sent this e-mail message. Notice the **From**, **To**, and **Subject** headers.
6. Close the Notepad window.
7. Turn off 6235A-NY-SQL-01 and delete changes.

Results: After this exercise, the native transactions have been implemented in the package.

Module 8: Configuring and Deploying Packages

Lab: Deploying Packages

Exercise 1: Creating a Package Configuration

► **Task 1: Configure the lab environment**

1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
2. Log on to **NY-SQL-01** as **Student** using the password **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. The Windows Explorer window opens.
5. Browse to the **E:\MOD08\Labfiles\Starter** folder, and then double-click **Setup.cmd**.
6. The script executes in a console window. Ignore any error messages displayed in the console window. Wait for the console window to close before proceeding to the next step.
7. On the **Start** menu, point to **Administrative Tools** and then click **Services**.
8. The **User Account Control** dialog box appears. Click **Continue**.
9. The Services window opens. In the right pane, click **Simple Mail Transfer Protocol (SMTP)** and then click the **Start** button.
10. Close Services.
11. Keep the Windows Explorer window open. You will use it later on.

► **Task 2: Open an SSIS solution**

1. Click Start | All Programs | Microsoft SQL Server 2008 | SQL Server Business Intelligence Development Studio.
2. Right-click the **Start Page** tab, and then click **Close**.
3. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. The **Open Project** dialog box appears. Browse to the E:\MOD08\Labfiles\Starter\SSIS_sol8 folder, and then double-click **SSIS_sol8.sln**.
5. In Solution Explorer, double-click **Employees.dtsx** beneath the **SSIS Packages** node.

► **Task 3: Add two variables to the package**

1. Click the control flow design surface to ensure that no containers or tasks are selected.
2. On the **SSIS** menu, click **Variables**.
3. In the **Variables** pane, click the **Add Variable** button.
4. In the row for the new variable, in the **Name** field, type **BCCLine**.
5. Ensure that the **Scope** field shows **Employees** as the variable scope.
6. In the **Data Type** list, click **String**.
7. In the **Value** field, type **sqlserver@adventure-works.com**.
8. Click the **Add Variable** button.
9. In the row for the new variable, in the **Name** field, type **ToLine**.
10. Ensure that the **Scope** field shows **Employees** as the variable scope.
11. In the **Data Type** list, click **String**.
12. In the **Value** field, type **Administrator@adventure-works.com**.
13. Close the **Variables** pane.

► **Task 4: Configure property expressions for a task**

1. On the control flow design surface, right-click the **SEM Send e-mail on success Send Mail** task, and then click **Properties**.
2. In the **Properties** pane, in the **Expressions** field, click the **ellipsis** button.
3. The **Property Expressions Editor** dialog box appears. In the first row of the grid, in the **Property** list, click **BCCLine**.
4. In the **Expression** field for that row, type `@[User::BCCLine]`.
5. In the second row of the grid, in the **Property** list click **ToLine**.
6. In the **Expression** field for that row, type `@[User::ToLine]`.
7. Click **OK** to close the **Property Expressions Editor** dialog box.

► **Task 5: Add a package configuration**

1. Click the control flow design surface.
2. On the **SSIS** menu, click **Package Configurations**.
3. The **Package Configuration Organizer** dialog box appears. Select the **Enable package configurations** check box, and then click **Add**.
4. The **Package Configuration Wizard** wizard appears. Click **Next**.
5. On the Select Configuration Type page, verify that **XML configuration file** is selected in the **Configuration type** list.
6. In the **Configuration file name** field, type `E:\MOD08\Labfiles\Starter\Employees\SendMail.dtsConfig`, and then click **Next**.
7. On the Select Properties to Export page, expand the **BCCLine** node (beneath the **Variables** node), expand the **Properties** node, and then select the **Value** check box.
8. Expand the **ToLine** node, expand the **Properties** node, and then select the **Value** check box.
9. Click **Next**.

10. On the Completing the Wizard page, in the **Configuration name field**, type **SendMail**.
11. Review the settings in the **Preview** box, and then click **Finish**.
12. Click **Close** to close the **Package Configuration Organizer** dialog box.

► **Task 6: View the package configuration file**

1. In Windows Explorer navigate to the **E:\MOD08\Labfiles\Starter\Employees** folder.
2. Verify that Integration Services created the **SendMail.dtsConfig** file.
3. Double-click the **SendMail.dtsConfig** file to view the file in Internet Explorer.
4. The Windows Internet Explorer window opens. If the **Information Bar** dialog box appears, click **Close**.
5. View the XML data to verify that it contains the **BCCLine** and **ToLine** properties and their values.
6. Close Internet Explorer.

Results: After this exercise, you should have created a package configuration.

Exercise 2: Preparing a Package for Deployment

- ▶ **Task 1: Set package properties and remove unnecessary objects**
 1. In the SQL Server Business Intelligence Development Studio, right-click the control flow design surface, and then click **Properties**.
 2. In the **Properties** pane, in the **PackagePriorityClass** list, click **AboveNormal**.
 3. In the control flow design surface, right-click the **DFT Test package Data Flow** task, and then click **Delete**.
 4. The **Confirm Deletion of Tasks** dialog box appears. Click **Yes**.
 5. In the **Connection Managers** pane, right-click **AdventureWorks2008_cm**, and then click **Delete**.
 6. The **Confirm Deletion of Tasks** dialog box appears. Click **Yes**.
 7. In the **Connection Managers** pane, right-click **AdventureWorksDW2008_cm**, and then click **Delete**.
 8. The **Confirm Deletion of Tasks** dialog box appears. Click **Yes**.
 9. On the toolbar, click the **Save All** button.

- ▶ **Task 2: Configure the deployment properties and build the package**
 1. In Solution Explorer, right-click **SSIS_proj8**, and then click **Properties**.
 2. The **SSIS_proj8 Property Pages** dialog box appears. In the navigation pane, click **Deployment Utility**.
 3. On the Deployment Utility page, in the **CreateDeploymentUtility** list, click **True**.
 4. Click **OK**.
 5. On the **Build** menu, click **Build SSIS_proj8**.
 6. Verify that SSIS Designer displays the **Build succeeded** message in the bottom left corner of the window status bar.

► **Task 3: View the deployment utility manifest**

1. In Windows Explorer, navigate to the **E:\MOD08\Labfiles\Starter\SSIS_sol8\SSIS_proj8\bin\Deployment** folder.
2. Verify that the folder contains the **SSIS_proj8.SSISDeploymentManifest** file.
3. Right-click the **SSIS_proj8.SSISDeploymentManifest** file, and then click **Open With**.
4. The **Open With** dialog box appears. If necessary, expand **Other Programs**.
5. In the **Other Programs** section, click **Internet Explorer**.
6. Clear the **Always use the selected program to open this kind of file** check box, and then click **OK**.
7. The Internet Explorer window opens. If the **Information Bar** dialog box appears, click **Close**.
8. In Internet Explorer, verify that the XML data references the **SendMail.dtsConfig** configuration file.
9. Close Internet Explorer.

Results: After this exercise, the new package has been prepared for deployment.

Exercise 3: Deploying a Package

► Task 1: Use the Package Installation Wizard to deploy the package

1. In Windows Explorer, in the E:\MOD08\Labfiles\Starter\SSIS_sol8\SSIS_proj8\bin\Deployment folder, double-click SSISDeploymentManifest.
2. The **Package Installation Wizard** wizard appears. Click **Next**.
3. On the Deploy SSIS Packages page, click **SQL Server deployment**, select the **Validate packages after installation** check box, and then click **Next**.
4. On the **Specify Target SQL Server** page, verify that **(local)** is selected in the **Server name** list and that **Use Windows Authentication** is selected as the authentication type.
5. In the **Package path** field, type **/Data Collector**.
6. Select the **Rely on server storage for encryption** check box, and then click **Next**.
7. On the Select Installation Folder page, verify that the folder specified in the **Folder** field is C:\Program Files\Microsoft SQL Server\100\DTS\Packages\SSIS_proj8, and then click **Next**.
8. On the Confirm Installation page, click **Next**.
9. The wizard will confirm the installation before displaying the next page of the wizard.
10. On the Configure Packages page, in the **Configurations** box, expand the **Property** node.
11. Verify that the configuration file includes the **BCCLine** and **ToLine** package variables and their values, and then click **Next**.
12. On the Packages Validation page, confirm that all the tasks and containers were successfully validated, and then click **Next**.
13. On the Finish the Package Installation Wizard page, review the deployment information, and then click **Finish**.

► **Task 2: Verify that the package has been deployed**

1. Click Start | All Programs | Microsoft SQL Server 2008 | SQL Server Management Studio.
2. The Microsoft SQL Server Management Studio window opens. The **Connect to Server** dialog box appears. Click **Connect**.
3. In Object Explorer, click **Connect**, and then click **Integration Services**.
4. The **Connect to Server** dialog box appears. In the **Server name** field, type **localhost**, and then click **Connect**.
5. In Object Explorer, expand **Stored Packages** | **MSDB** | **Data Collector**.
6. Verify that the **Employees** package has been deployed to the local instance of **Integration Services**.

► **Task 3: Verify that the configuration file has been deployed**

1. In Windows Explorer, navigate to the C:\Program Files\Microsoft SQL Server\100\DTS\Packages\SSIS_proj8 folder.
2. Verify that the file **SendMail.dtsConfig** has been deployed to the folder.

► **Task 4: Run the package**

1. In the Microsoft SQL Server Management Studio, in the Object Explorer, right-click the **Employees** package, and then click **Run Package**.
2. The **Execute Package Utility** dialog box appears. Review the settings used to run the **Employees** package, and then click **Execute**.
3. The Package Execution Progress window opens. View the execution progress in the **Package Execution Progress** dialog box.
4. Examine the results and make sure that the validation completes successfully.
5. Click **Close** to close the **Package Execution Progress** dialog box.
6. Click **Close** to close the **Execute Package Utility** dialog box.
7. Turn off 6235A-NY-SQL-01 and delete changes.

Results: After this exercise, the new package has been deployed and tested.

Module 9: Managing and Securing Packages

Lab: Managing and Securing Packages

Exercise 1: Importing a Package

► **Task 1: Configure the lab environment**

1. In the Lab Launcher, next to 6235A-NY-SQL-01, click **Launch**.
2. Log on to NY-SQL-01 as **Student** using the password **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. Browse to the E:\MOD09\Labfiles\Starter folder, and then double-click **Setup.cmd**.
5. The script executes in a console window. Ignore any error messages displayed in the console window. Wait for the console window to close before proceeding to the next step.
6. On the **Start** menu, point to **Administrative Tools**, and then click **Services**.
7. The **User Account Control** dialog box appears. Click **Continue**.
8. The Services window opens. In the right pane, click **Simple Mail Transfer Protocol (SMTP)**, and then click the **Start** button.
9. Close Services.
10. Click **Start | All Programs | Microsoft SQL Server 2008**.
11. Right-click **SQL Server Management Studio**, and then click **Run as administrator**.
12. The **User Account Control** dialog box appears. Click **Continue**.
13. The Microsoft SQL Server Management Studio window opens. The **Connect to Server** dialog box appears. Click **Connect**.
14. On the Object Explorer toolbar, click **Connect**, and then click **Integration Services**.
15. The **Connect to Server** dialog box appears. In the **Server name** field, type **localhost**, and then click **Connect**.

► **Task 2: Create a folder structure for packages**

1. In Object Explorer, expand **Stored Packages | MSDB**.
2. Right-click the **MSDB** folder, and then click **New Folder**.
3. The **Create New Folder** dialog box appears. In the **Enter the name of the new folder** field, type **HumanResources**, and then click **OK**.
4. Right-click the **MSDB** folder, and then click **New Folder**.
5. The **Create New Folder** dialog box appears. In the **Enter the name of the new folder** field, type **Sales**, and then click **OK**.
6. Right-click the **HumanResources** folder, and then click **New Folder**.
7. The **Create New Folder** dialog box appears. In the **Enter the name of the new folder** field, type **Employees**, and then click **OK**.

► **Task 3: View the data about the folders in the msdb database**

1. On the toolbar, click **New Query**.
2. If the **Connect to Server** dialog box appears, click **Connect**.
3. On the toolbar, in the **Available Databases** list, click **msdb**.
4. In the query editor, type the following SQL statement:

```
SELECT * FROM dbo.sysssispackagefolders
```
5. On the toolbar, click **Execute**.
6. View the query results in the lower pane of the query editor. Notice that the results include a row for each folder that you created.

► **Task 4: Import the package into a folder**

1. In Object Explorer, right-click the **Sales** node (within the **Stored Packages | MSDB** folder structure), and then click **Import Package**.
2. The **Import Package** dialog box appears. In the **Package location** list, click **File System**.
3. Click the **browse** button next to the **Package path** field.
4. The **Load Package** dialog box appears. Browse to the **E:\MOD09\Labfiles\Starter\SSIS_sol9\SSIS_proj9\bin** folder, click **Demographics.dtsx**, and then click **Open**.
5. Confirm that the **Package name** field displays the package name **Demographics**, and then click **OK**.
6. In Object Explorer, expand **Stored Packages | MSDB | Sales**.
7. Verify that the **Demographics** package has been added beneath the **Sales** node in Object Explorer.

► **Task 5: View the data about the package in the msdb database**

1. In the query tab change the SQL statement to the following:

```
SELECT * FROM dbo.sysssispackages
```
2. On the toolbar, click **Execute**.
3. View the query results. The table should contain a row for the Demographics package.
4. Keep SQL Server Management Studio open. You will use it in the next exercise.

Results: After this exercise, the package has been imported and tested.

Exercise 2: Configuring, Executing, and Monitoring a Package

► Task 1: Add a configuration file to the package execution

1. In Object Explorer, right-click the **Demographics** package, and then click **Run Package**.
2. The **Execute Package Utility** dialog box appears. On the Configurations page, click **Add**.
3. The **Open** dialog box appears. Browse to the **E:\MOD09\Labfiles\Starter\Employees** folder, click **SendMail.dtsConfig**, and then click **Open**.

► Task 2: Configure the package to fail on validation warnings

1. On the Execution Options page, select the **Fail the package on validation warnings** check box.
2. Verify that the **Validate package without executing** check box and the **Enable package checkpoints** check box are cleared.
3. Verify that the **Maximum concurrent executables** list is set to **-1**.

► Task 3: Configure console logging to log the source name and message

1. On the Reporting page, in the **Console logging** section, select the **Source name** check box and the **Message** check box.
2. In the **Console events** section, verify that only the **Verbose** check box is selected.

► Task 4: Disable event handlers

1. On the Set Values page, in the **Property Path** field of the first row of the **Properties** grid, type **\Package.Properties[DisableEventHandlers]**.
2. In the **Value** field of the first row, type **True**.

► **Task 5: Run the package**

1. Click **Execute**.
2. The Package Execution Progress window opens.
3. Switch to the SQL Server Management Studio window.
4. In Object Explorer, right-click the **Running Packages** folder, and then click **Refresh**.
5. Expand the **Running packages** node.
6. Notice that the **Demographics** package appears in the **Running packages** list.
7. Click the **Refresh** button as necessary to view updated information about the running package. The report is available only as long as the package is running. If the package has already finished, close the Package Execution Progress window, and click **Execute** in the **Execute Package Utility** dialog box again.

► **Task 6: View the package execution progress**

1. In the Package Execution Progress window, view the execution progress.
2. Wait for the package execution to complete and the **Close** button to become active.
3. Verify that all validations completed successfully.

Note: It may take as long as 10 minutes to complete the job.

4. Click **Close** to close the Package Execution Progress window.
5. Click **Close** to close the **Execute Package** utility.
6. In Object Explorer, right-click the **Running Packages** folder, and then click **Refresh**.
7. Expand the **Running packages** node.
8. Notice that the **Demographics** package no longer appears in the **Running packages** list.

► **Task 7: Verify that an email message was sent**

1. In Microsoft Windows Explorer, navigate to the C:\inetpub\mailroot\Drop folder.
2. If that folder is empty, navigate to the C:\inetpub\mailroot\Queue folder.
3. In the details pane, double-click the **Internet E-Mail Message** file.
4. The **Windows** dialog box appears. Click **Select a program from a list of installed programs**, and then click **OK**.
5. The **Open With** dialog box appears. Expand **Other Programs**, click **Notepad**, and then click **OK**.
6. The Notepad window opens.
7. Verify that the **SMT Send e-mail on success Send Mail** task has sent this e-mail message. Notice the **From**, **To**, and **Subject** headers.
8. Close the Notepad window.

Results: After this exercise, the package has been configured, executed, and then monitored.

Exercise 3: Scheduling a Package

► **Task 1: Create a SQL Server Agent job to run a package**

1. In Object Explorer, expand the **SQL Server Agent** node.
2. Right-click the **Jobs** folder, and then click **New Job**.
3. The **New Job** dialog box appears. On the General page, in the **Name** field, type **Run Demographics**.
4. Verify that **NY-SQL-01\Student** is displayed in the **Owner** field, **[Uncategorized (Local)]** is selected in the **Category** list, and the **Enabled** check box is selected.
5. On the Steps page, click **New**.
6. The **New Job Step** dialog box appears. In the **Step name** field, type **Run package**.
7. In the **Type** list, click **SQL Server Integration Services Package**.
8. In the **Package source** list, click **SSIS Package Store**.

9. In the **Server** list, click **NY-SQL-01**.
10. Verify that **Use Windows Authentication** is selected in the **Log on to the server** section.
11. In the **Package** field, click the **ellipsis** button.
12. The **Select an SSIS Package** dialog box appears. Browse to the **\MSDB\Sales** folder.
13. Click the **Demographics** package, and then click **OK**.
14. Click the **Configurations** tab.
15. Click **Add**.
16. The **Browse for File - NY-SQL-01** dialog box appears. Browse to the **E:\MOD09\Labfiles\Starter\Employees** folder.
17. Click **SendMail.dtsConfig**, and then click **OK**.
18. Click **OK** to close the **New Job Step** dialog box.
19. On the Schedules page of the **New Job** dialog box, click **New**.
20. The **New Job Schedule** dialog box appears. In the **Name** field, type **Run once**.
21. In the **Schedule type** list, click **One time**.
22. Set the time in the **Time** field to two minutes later than the current time.
23. Click **OK** to close the **New Job Schedule** dialog box, and then click **OK** to close the **New Job** dialog box.

► Task 2: View job activity

1. In Object Explorer, right-click the **Job Activity Monitor** node, and then click **View Job Activity**.
2. The **Job Activity Monitor - NY-SQL-01** dialog box appears. View the details about the **Run Demographics** job.
3. Shortly after the time that the job is scheduled to start, click the **Refresh** button and verify that the **Status** is shown as **Executing**.

4. Click **Refresh** every 30 seconds or so until the **Status** is shown as **Idle**.
- Note:** It may take as long as 10 minutes to complete the job.
5. Verify that the details of the job's **Last Run Outcome** and **Last Run** are shown.
 6. Click **Close**.

► **Task 3: View job history.**

1. In Object Explorer, expand the **Jobs** folder, right-click the **Run Demographics** job, and then click **View History**.
2. The **Log File Viewer - NY-SQL-01** dialog box appears. View the details about the **Run Demographics** job. View additional information by clicking the plus sign at the front of the row for that job.
3. Click **Close**.

Results: After this exercise, the package has been scheduled for execution.

Exercise 4: Securing a Package

► **Task 1: Open an Integration Services solution**

1. Click **Start | All Programs | Microsoft SQL Server 2008 | SQL Server Business Intelligence Development Studio**.
2. Right-click the **Start Page** tab, and then click **Close**.
3. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. The **Open Project** dialog box appears. Browse to the **E:\MOD09\Labfiles\Starter\SSIS_sol9** folder.
5. Click **SSIS_sol9.sln**, and then click **Open**.
6. In Solution Explorer, double-click **Employees.dtsx** beneath the **SSIS Packages** node.

► **Task 2: Encrypt a package with a user key**

1. Right-click on the control flow design surface, and then click **Properties**.
2. In the **Properties** pane, in the **ProtectionLevel** list, click **EncryptAllWithUserKey**.
3. On the toolbar, click the **Save All** button.
4. On the **Build** menu, click **Build SSIS_proj9**.
5. Wait for the **Build succeeded** message in the bottom left corner of SSIS Designer, and then close Business Intelligence Development Studio.

► **Task 3: Import the package**

1. In Microsoft SQL Server Management Studio, in Object Explorer, expand **Stored Packages** | **MSDB** | **HumanResources**. Right-click the **Employees** node, and then click **Import Package**.
2. The **Import Package** dialog box appears. In the **Package location** list, click **File System**.
3. Click the **browse** button next to the **Package path** field.
4. The **Load Package** dialog box appears. Browse to the **E:\MOD09\Labfiles\Starter\SSIS_sol9\SSIS_proj9\bin** folder, click **Employees.dtsx**, and then click **Open**.
5. Verify that the **Package name** field displays the package name **Employees**.
6. Click **OK**.
7. In Object Explorer, expand **Stored Packages** | **MSDB** | **HumanResources** | **Employees**.
8. Verify that the **Employees** package has been added beneath the **Employees** folder in Object Explorer.

► **Task 4: Configure the package roles for the package**

1. In Object Explorer, right-click the **Employees** package, and then click **Package Roles**.
2. The **Package Roles** dialog box appears. Verify that <Default: db_dtsadmin, db_dtsoperator, creator of the package> is selected in the **Reader Role** list.
3. Verify that <Default: db_dtsadmin, creator of the package> is selected in the **Writer Role** list.
4. Click **OK**.

► **Task 5: Run the package**

1. Right-click the **Employees** package, and then click **Run Package**.
2. The **Execute Package Utility** dialog box appears. In the Execute Package utility, review the settings used to run the **Employees** package. Use the default settings for all options.
3. Click **Execute**.
4. The **Package Execution Progress** dialog box appears. View the execution progress in the Package Execution Progress window.
5. Click **Close** to close the **Package Execution Progress** dialog box.
6. Click **Close** to close the **Execute Package Utility** dialog box.
7. Turn off 6235A-NY-SQL-01 and delete changes.

Results: After this exercise, the package has been secured.