

Microsoft®

OFFICIAL MICROSOFT LEARNING PRODUCT

6236A

**Implementing and Maintaining
Microsoft® SQL Server® 2008
Reporting 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, Excel, Internet Explorer, Microsoft Press, MSDN, PowerPoint, SharePoint, SQL Server, Visual Basic, Visual Studio, Windows, and Windows Server 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.

Technical Reviewer: Tan Jian Bo

Product Number: 6236A

Part Number: X15-01570

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 – Lead 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.

Jerry Knowles – Content Developer

Mr. Knowles joined Aeshen in 2008 as an Application Analyst. He has worked in Information Technology since 1989 as an instructor, application developer, SQL database administrator, and consultant.

Paul Pardi – Content Developer

Paul has worked professionally in software development for over 15 years. He spent 10 years at Microsoft as a Software Development Manager and has worked as a manager, developer, consultant, and classroom trainer. Paul co-authored the book Access 2003 Inside Track and has written numerous technical articles for an online technical magazine.

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.

Tan Jian Bo – Technical Reviewer

Jian Bo is Microsoft Certified Trainer since 2002. He has been providing technical consultancy service for over 10 years including 6 years as solution architect for various projects and 8 years as various roles in training industry.

Contents

Module 1: Introduction to Microsoft SQL Server Reporting Services

Lesson 1: Overview of SQL Server Reporting Services	1-3
Lesson 2: Installing Reporting Services	1-10
Lesson 3: Reporting Services Tools	1-19
Lab: Using Reporting Services Tools	1-25

Module 2: Authoring Basic Reports

Lesson 1: Creating a Basic Table Report	2-3
Lesson 2: Formatting Report Pages	2-11
Lesson 3: Calculating Values	2-16
Lab: Authoring Basic Reports	2-24

Module 3: Enhancing Basic Reports

Lesson 1: Interactive Navigation	3-3
Lesson 2: Displaying Data	3-11
Lab: Enhancing a Report	3-24

Module 4: Manipulating Data Sets

Lesson 1: Defining Report Data	4-3
Lesson 2: Using Parameters and Filters	4-7
Lesson 3: Using Parameter Lists	4-12
Lab: Manipulating Data Sets	4-21

Module 5: Using Report Models

Lesson 1: Creating Report Models	5-3
Lesson 2: Using Report Builder	5-10
Lab: Working with Report Models	5-19

Module 6: Publishing and Executing Reports

Lesson 1: Publishing Reports	6-3
Lesson 2: Executing Reports	6-9
Lesson 3: Creating Cached Instances	6-14
Lesson 4: Creating Snapshots and Report History	6-20
Lab: Publishing and Executing Reports	6-30

Module 7: Using Subscriptions to Distribute Reports

Lesson 1: Introduction to Report Subscriptions	7-3
Lesson 2: Creating Report Subscriptions	7-8
Lesson 3: Managing Report Subscriptions	7-17
Lab: Implementing Subscriptions	7-24

Module 8: Administering Reporting Services

Lesson 1: Reporting Services Administration	8-3
Lesson 2: Performance and Reliability Monitoring	8-11
Lesson 3: Administering Report Server Databases	8-23
Lesson 4: Security Administration	8-32
Lesson 5: Upgrading to Report Services 2008	8-46
Lab: Administering Reporting Services	8-49

Module 9: Programming Reporting Services

Lesson 1: Querying for Server Information Using a Web Service	9-3
Lesson 2: Automating Report Management	9-9
Lesson 3: Rendering Reports	9-13
Lesson 4: Creating Custom Code	9-22
Lab: Programming Reporting Services	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 a Reporting Services solution in an organization. The course discusses how to use the Reporting Services development tools to create reports, and how to use the Reporting Services management and administrative tools to manage a Reporting Services solution.

Audience

This course is intended for information technology (IT) professionals and developers who need to implement reporting solutions by using Microsoft SQL Server 2008 Reporting Services.

Student Prerequisites

This course requires that you meet the following prerequisites:

- Exposure to creating reports in Microsoft Access or other third-party report products, such as Crystal Reports.
- Conceptual understanding of the push and pull distribution/subscription paradigm.
- Experience navigating the Microsoft Windows Server environment.
- Experience with Windows services (starting and stopping).
- Experience creating service accounts and permissions.
- 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 Reporting Services and its components.
- Create a Reporting Services report.
- Enhance a Reporting Services report.
- Create and manipulate data sets.
- Use report models to implement reporting for business users.
- Configure report publishing and execution settings.
- Implement subscriptions for reports.
- Administer Reporting Services.
- Implement custom Reporting Services applications.

Course Outline

This section provides an outline of the course:

Module 1: Introduction to Microsoft SQL Server Reporting Services

This module covers the role that Reporting Services plays in an organization's reporting life cycle, the key features offered by Reporting Services, and the components that make up the Reporting Services architecture.

Module 2: Authoring Basic Reports

This module covers the fundamentals of report authoring, including configuring data sources and data sets, creating tabular reports, summarizing data, and applying basic formatting.

Module 3: Enhancing Basic Reports

This module covers navigational controls and some additional types of data regions, and how to use them to enhance a basic report.

Module 4: Manipulating Data Sets

This module covers data sets to a greater depth, including the use of alternative data sources and interacting with a data set through the use of parameters. It also covers how to dynamically modify the data set underlying a data region by allowing parameters to be sent to the underlying query, as well as will learn to use best practices to implement static and dynamic parameter lists when interacting with queries and stored procedures.

Module 5: Using Report Models

This module covers how to create a report model so that business users can create their own reports without using the full Report Designer development environment. It also covers will also learn how to use Report Builder to create a report from a report model.

Module 6: Publishing and Executing Reports

This module covers the various options you can use to publish reports to the report server and execute them.

Module 7: Using Subscriptions to Distribute Reports

This module covers how to implement subscriptions so that you can distribute reports either automatically by e-mail or by publishing reports to a shared folder.

Module 8: Administering Reporting Services

This module covers how to administer the Reporting Services server, how to monitor and optimize the performance of the report server, how to maintain the Reporting Services databases, and how to keep the system secure.

Module 9: Programming Reporting Services

This module covers how to query Reporting Services information programmatically and how to automate report management tasks. Students will also learn how to render reports without relying on Report Manager, and how to extend the feature set of a report server by creating custom code.

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 fingertips 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 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
6236A-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 Microsoft SQL Server Reporting Services

Contents:

Lesson 1: Overview of SQL Server Reporting Services	1-3
Lesson 2: Installing Reporting Services	1-10
Lesson 3: Reporting Services Tools	1-19
Lab: Using Reporting Services Tools	1-25

Module Overview

- Overview of SQL Server Reporting Services
- Installing Reporting Services
- Reporting Services Tools

You will be introduced to the role that Microsoft® SQL Server® Reporting Services plays in an organization's reporting life cycle, the key features offered by Reporting Services, and the components that make up the Reporting Services architecture.

With Reporting Services, you will be able to more effectively create Reports and output that your organization needs with more flexibility and less time for delivery.

Lesson 1

Overview of SQL Server Reporting Services

- Top New Features
- The Reporting Lifecycle
- Highlights of Reporting Services
- Reporting Services Scenarios

Reporting Services can query and display data from any of the multiple databases that exist in an organization today. In this lesson we will examine the methods available to attach to those databases and setup queries to extract the data.

With Reporting Services you can now create even more graphical and free form reports that in previous versions of SQL Server.

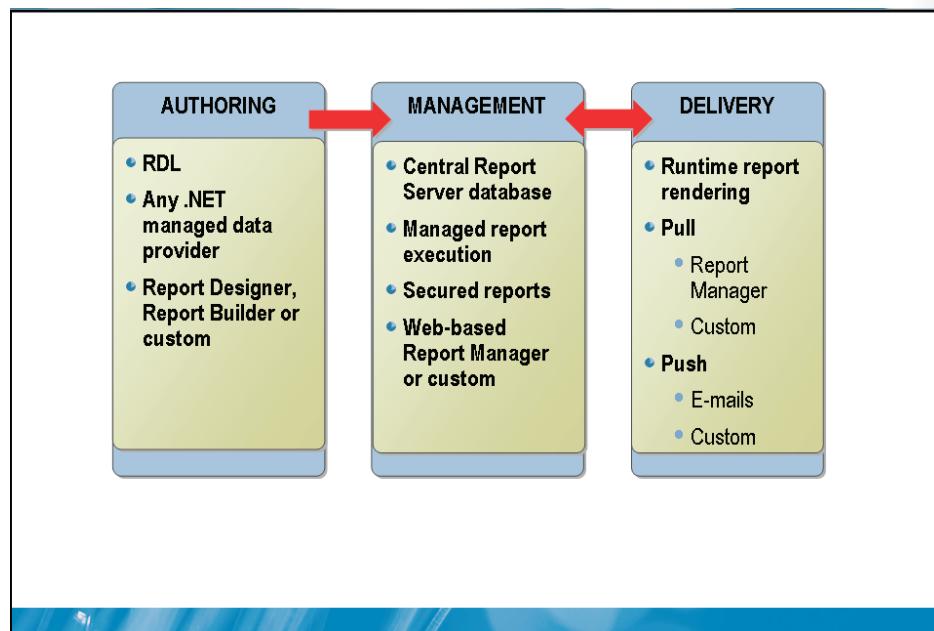
Top New Features

- Create reports with any structure using the unique data format of Tablix
- Benefit from enhanced performance and scalability and reach any number of users
- Render reports to Microsoft Office Word format
- Integrate Reporting Services with Microsoft Office SharePoint Services for central delivery and management of business insight
- Build reports with richly formatted text
- Display data graphically with enhanced visualization capabilities

Key Points

SQL Server 2008 Reporting Services introduce Tablix, Chart, and Gauge data regions. It also introduces support for richly formatted text, new data source types, and Report Builder 2.0, which offers many new features, like enhanced data layout and visualization, in an authoring environment similar to that of the Microsoft Office 2007 family. Finally, this topic describes incremental changes to authoring tools and the Report Definition Language (RDL) that allow a report author to take full advantage of new processing features.

The Reporting Lifecycle



Key Points

The process of report creation, administration, and dissemination (known collectively as the reporting lifecycle) can be broken down into three stages: authoring, management, and delivery.

- In the authoring stage, the report author defines the report data and presentation.
- RDL provides an Extensible Markup Language (XML) representation of the report definition.
- In the management stage, the author publishes the report to a central location where the report manager can administer the report security and execution schedule.
- After publishing reports to the Report Server database, you can create an execution schedule and set security options for the report.

- In the delivery stage, you can make reports available in a number of rendered formats. Reports can also be delivered on demand or scheduled based on an event.
- At run time, Reporting Services processes or renders the presentation style of the report.

Highlights of Reporting Services

Lifecycle Stage	Highlight
Authoring	<ul style="list-style-type: none">● Wide range of supported data sources● Open report authoring options● Flexible report designs
Management	<ul style="list-style-type: none">● Parameterized reports● Execution properties● Report scheduling and history● Role-based security
Delivery	<ul style="list-style-type: none">● Range of rendering options● Flexible and extensible delivery

Key Points

As a component of SQL Server 2008, Reporting Services provides a number of key features. In this topic, you will explore some of those features in the context of the reporting lifecycle.

In the authoring stage of the reporting lifecycle, Reporting Services supports the following important features and functionality:

- Access to a wide range of supported data sources using ADO.NET managed providers.
- You can create RDL files using Report Designer, Report Builder, or any XML compatible editor.

- Data regions provide a wide range of formatting options during report design. Supported data regions include Tablix, List, Gauge and Chart.
- With parameterized reports, users can specify values such as month or salesperson to generate specific reports, or they can use default values provided by the report.

Question: Which of these new and updated Data Region features could you see benefiting your organization?

Reporting Services Scenarios

Key Reporting Scenarios:

- Application Integration
- Embedding Reports in Custom Applications
- Redistribution Through SQL Server Express with Advanced Services
- Hosting Reports in Custom Dashboards and SharePoint Sites
- Building Custom Report Design and Report Management Tools
- Extending Reporting Services Functionality

Key Points

Reporting Services is an enterprise reporting solution that meets a wide range of implementation scenarios.

- Out-of-the-box tools and applications are available for expert report designers as well as information workers who need an easy way to create their own reports and explore business data.
- If you are developer, you can integrate Reporting Services functionality within a custom application, or extend it to support the type of functionality you require.

Lesson 2

Installing Reporting Services

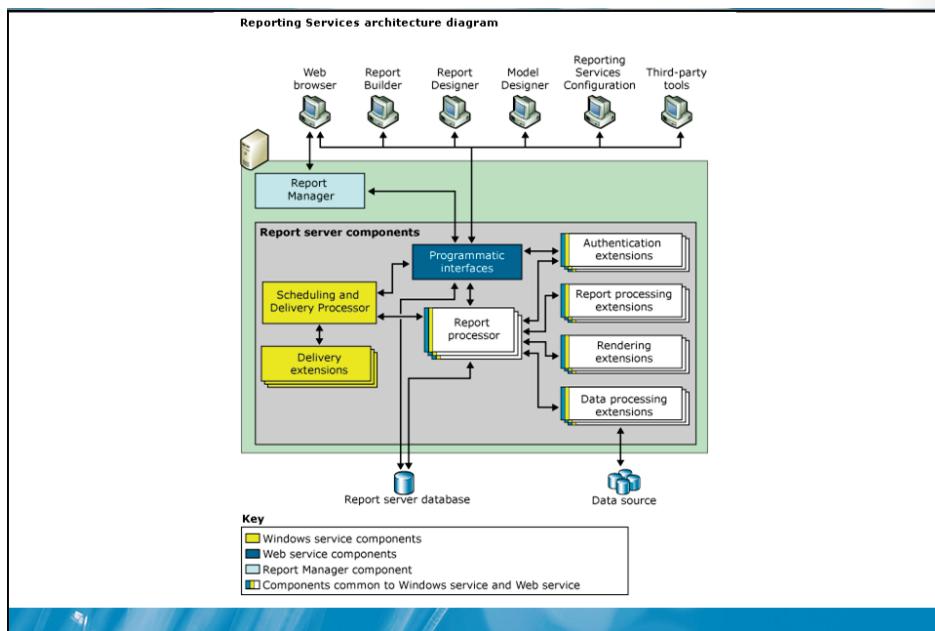
- Reporting Services Architecture
- Reporting Services Installation
- Remote Installation
- Installation Scenarios
- SharePoint Integration

Reporting Services is a collection of server and client applications used for business reporting. You can install server components on a single computer or on multiple computers. You can install the client components alongside server components or on separate workstations.

Choosing the default configuration when installing Reporting Services by using the SQL Server setup program automates the entire configuration process for Reporting Services.

It is important to know the requirements before installing Reporting Services to eliminate any possible trouble shooting issues.

Reporting Services Architecture



Key Points

Reporting Services is a .NET framework-based platform that includes a comprehensive set of tools that you can use to integrate a reporting solution into any centrally-managed technical environment.

- Report Server contains a Web service that includes several subcomponents to manage report processing: data processing, report rendering, security policies, and report delivery. The Report Manager Web site provides a graphical interface to the Web service.
- The Report Server database is a SQL Server database that stores the information that Report Server uses to manage reports and resources. The Report Server database performs stateless storage of all information. This means that Report Server treats each interaction request solely on information in that request. You can have one or more Report Servers that all point to the same Report Server database.

- Programmatic interfaces allow you to create custom client applications that utilize Reporting Services, or alternatively you can enhance the reports system by adding new features.
- Through a Simple Object Access Protocol (SOAP) API over HTTP, you can create custom tools for any stage of the reporting lifecycle—authoring, management, or delivery.

Question: What is the advantage of having the Web service built into the architecture instead of being an external application?

Reporting Services Installation

- Installed through SQL Server Setup
- Native Mode Installation
- SharePoint Integrated Mode
- Files Only Installation
- Multiple instances supported

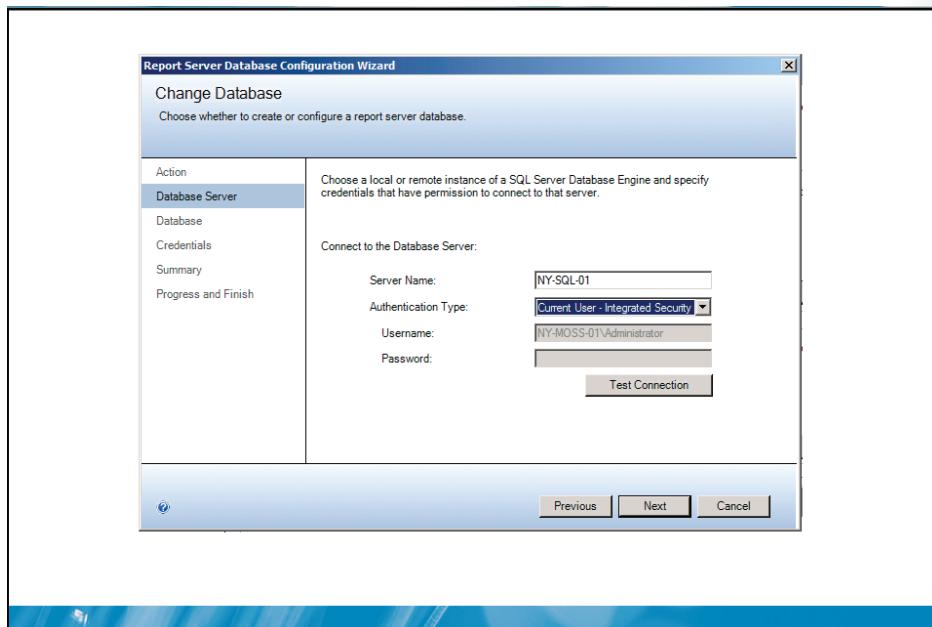
Key Points

Installing Reporting Services will involve modifications to a server computer and potentially client computers, depending on the client requirements.

- Reporting Services is installed as part of the SQL Server 2008 setup when selected as a requested component. You can either install Reporting Services on the local computer using a default installation or create a “files-only” installation that requires the Reporting Services Configuration Tool to complete the installation.
- You can install multiple instances of Reporting Services on the same computer. This is useful if you want to host different content for specific sites, each with its own report server database, configuration files, and virtual directories.

- To install a Report Server, your server needs the following features:
 - .NET Framework installed
 - Access to an SMTP Server for e-mail delivery of reports (optional)
 - Client computers need the following features:
 - A Web browser for viewing reports and interacting with Report Manager. Microsoft Windows Internet Explorer® 6.0 and later.
 - .NET Framework to support Business Intelligence Development Studio (BIDS) or Report Builder for designing reports (optional).

Remote Installation



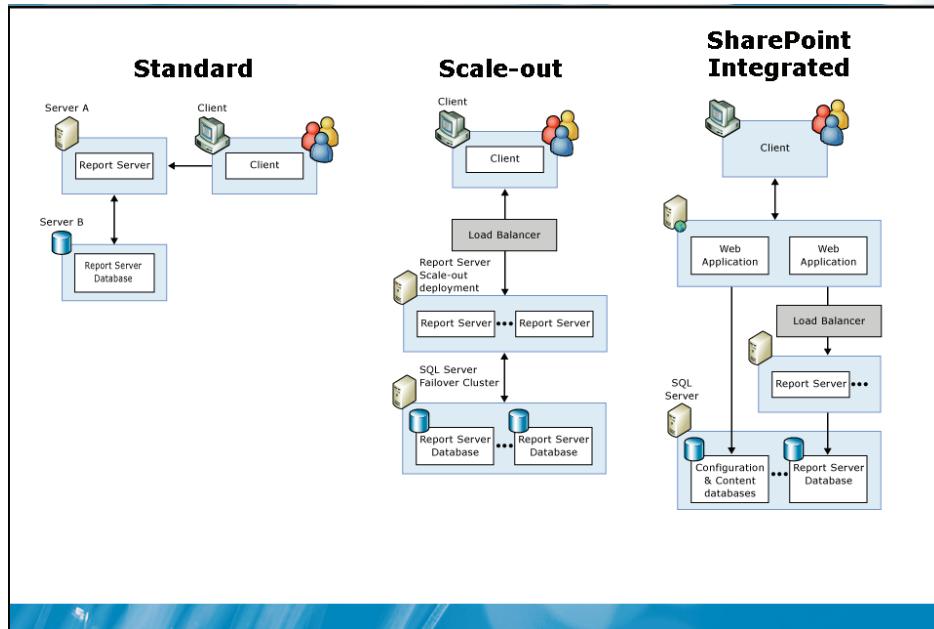
Key Points

SQL Server provides a single Setup program that installs all components, including Reporting Services. Using Setup, you can install Reporting Services with or without other SQL Server components on a single computer or remotely to a server that does not host any SQL based databases.

- Installs the Report Server service which includes the Report Server Web service and Report Manager. Also installs Report Builder, Reporting Services Configuration tool, and the Reporting Services command line utilities.
- Installs SQL Server Management Studio used to configure the report server, manage jobs, manage shared schedules, and configure role definitions.
- Installs the Visual Studio design environment that provides Report Designer and Model Designer for building Reporting Services projects.

Question: Now that you know more about Remote Installation, do you think you would use this feature in your own organization?

Installation Scenarios

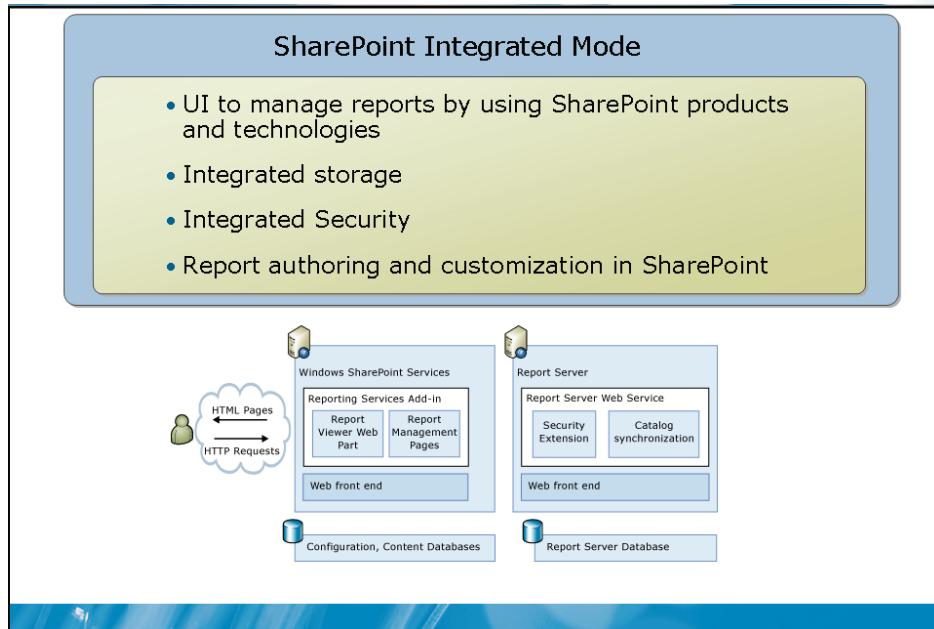


Key Points

When you install Reporting Services, you can consider four different installation scenarios to best suit your environment.

- A single server installation places the Report Server, Report Manager, and Report Server databases on the same physical server.
- A remote database server installation places the Report Server and Report Manager on a Web server with the Report Server databases on their own remote server.
- SharePoint integrated mode, where a report server is deployed as part of a SharePoint products and technologies server farm.
- SharePoint: Native mode, including native mode with SharePoint Web Parts, where a report server runs as an application server that provides all processing and management capability exclusively through Reporting Services components.

SharePoint Integration



Key Points

Reporting Services supports two levels of integration with SharePoint products and technologies. Full integration is supported through the SharePoint integrated mode deployment scenario. Partial integration is supported through a pair of Web Parts that you install on a SharePoint site and point to a remote report server instance.

- SharePoint integrated mode enables Reporting Services to integrate with the SharePoint databases and security model.
- Partial integration is supported through the Report Explorer and Report Viewer Web Parts that were first introduced in SQL Server 2000 Reporting Services Service Pack 2. These Web Parts continue to be available and enable you to select and view reports from a report server that is configured to use native mode.
- Provides a user interface (UI) to administer, secure, manage, view, and deliver reports by using SharePoint products and technologies. Reports, data sources, and data models are stored, accessed, and managed in a SharePoint library.

- Users publish or upload reports, models, and data sources to a SharePoint library.
- Publish reports, models, and data sources directly to a SharePoint library by uploading them in SharePoint or from Report Designer or Model Designer.

Question: What advantage would your organization see with SharePoint Integration?

Lesson 3

Reporting Services Tools

- Reporting Services Developer Tools
- Reporting Services Management Tools

With Reporting Services, you can create interactive, tabular, graphical, or free-form reports from relational, multidimensional, or XML-based data sources. You can publish reports, schedule report processing, or access reports on-demand. Reporting Services also enables you to create on-demand reports based on predefined models, and to interactively explore data within the model.

With Tablix and Gauge, you can now expand your Reporting capabilities even further. These options will help your organizations reporting needs and provide even more options.

Reporting Services Developer Tools

Tool	Purpose
Model Designer	Define, edit, and publish report models that are used in Report Builder
Report Designer	Design, preview, and publish reports from within Visual Studio environment
Report Builder	Allows information workers to design, test, and publish reports using a report model

Key Points

Reporting Services provides various tools that you can use to develop, test, and upload reports by various members within an organization.

- With Report Designer, you can develop reports interactively in Business Intelligence Development Studio and then publish them to a Report Server when they are ready to be deployed. Report Designer is not a separate application, but is available when working with Reporting Services solutions in BI Development Studio.
- With Model Designer, you can create a report model that Report Builder users can use to help them build on-demand reports. A report model is a business description of the underlying database that describes the data in terms of entities, attributes, and relationships (roles). Like Report Designer, Model Designer is part of Business Intelligence Development Studio.

- With Report Builder, information workers can build reports without having to understand the underlying data source structures in the database.
- Report Builder 1.0 is part of the SQL Server 2008 installation. Report Builder 2.0 is available as a separate download as part of the SQL Server 2008 Feature Pack. Report Builder 2.0 supports more robust authoring, more types of data sources, and also supports the 2008 version of the Report Definition Language (RDL).

Reporting Services Management Tools

Tool	Purpose
Report Manager	View and manage reports using a Web browser
Reporting Services Configuration Tool	Provides a graphical interface for configuring a report server
Command Prompt Utilities	<p>Rsconfig.exe manages a report server connection to the report server database</p> <p>Rskeymgmt.exe manages encryption keys used by the report server</p> <p>Rs.exe runs scripts that perform automated management features</p>

Key Points

Reporting Services provides various tools to manage and configure the Report Server environment.

- Report Manager is a Web-based report access and management tool that you connect to through a Web browser. The tool consists of an ASP.NET Web site that uses the Report Server Web services to provide users with the ability to browse existing reports, upload new reports, and manage all aspects of a report, including execution properties, security, and subscriptions.
- The Reporting Services Configuration Tool provides a graphical interface that uses the Reporting Services Windows Management Instrumentation (WMI) to configure a report server.

- You can use three command prompt utilities to manually configure parts of a report server.
 - Rsconfig.exe manages a report server connection to the report server database.
 - Rskeymgmt.exe manages encryption keys used by the report server so that items such as connection strings are maintained securely.
 - Rs.exe runs Microsoft Visual Basic® .NET scripts that perform automated management features like copying data between report server databases, publishing reports, or creating items in a report server database.

Question: From what you have learned of these tools, which do you think will suit your organization best?

Demonstration: Using Reporting Services Tools

In this demonstration, you will see how to:

- Start Business Intelligence Development Studio
- Open a sample report

Question: Where are data sources defined for a report?

Lab: Using Reporting Services Tools

- Exercise 1: Exploring Report Designer
- Exercise 2: Exploring Report Manager

Logon information

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

Estimated time: 45 minutes

Exercise 1: Exploring Report Designer

Scenario

In this exercise, you have been asked to use Report Designer to examine and edit the Sales Order Detail report. The Sales Team needs you to alter various parameters and generate a report for them to view.

The main tasks for this exercise are as follows:

1. Open the AdventureWorks2008 solution.
2. Explore the Sales Order Detail report.
3. Alter the Sales Order Detail report.
4. Alter the Order Detail Table region.
5. Change report parameter available values.

► **Task 1: Open the AdventureWorks2008 solution**

- Using Business Intelligence Development Studio, open the **AdventureWorks Sample Reports** solution located in the E:\MOD01\Labfiles\Starter folder.

► **Task 2: Explore the Sales Order Detail report**

- Open and preview the **Sales Order Detail** report.

Notice that the report is a parameterized report showing the detail of a particular sales order.

- In **Design** view, notice that the report is made up of a few distinct areas:
 - A header consisting of an Adventure Works image on the left, and a **Sales Order** title on the right together with the report parameter
 - A rectangle containing the billing, shipping and order header details
 - A table region containing the order detail together with a footer containing totals

► **Task 3: Alter the Sales Order Detail report**

- Add a **Textbox** control to the report, just above the text box containing the **Sales Order** title, aligned to the left edge.
- Format the new text box to be right-aligned and bold.
- Set the Expression for the text box to: **=Fields!OrderDate.Value**.
- Set the **Format** property to **d**.

► **Task 4: Alter the OrderDetail table region**

- Set the Expression of the cell in the footer row of the **Discount** column of the **OrderDetail** table to: **=Fields!UnitPriceDiscount.Value**.
- Set the **Format** property for this cell to **p**.
- Preview the report to view your changes.

► **Task 5: Change report parameter available values**

- To edit report parameters, while viewing the **Design** tab, on the **Report Data** pane, expand **Parameters**.
- Change the **SalesOrderNumber** parameter. In the parameter properties, type the following values in both the labels and values:
 - SO50750
 - SO50751
 - SO50752
- Preview the report for the sales order number **SO50751**.

Results: After this exercise, you should have a basic understanding of altering data and setting values for reporting queries.

Exercise 2: Exploring Report Manager

Scenario

In this exercise, you will use the Report Manager to view the Product Catalog report. You will then edit the properties of the Sales Order report.

The main tasks for this exercise are as follows:

1. Start Report Manager.
2. Examine the Product Catalog report.
3. View the Company Sales report.
4. Export the Company Sales report to Excel.
5. Examine Company Sales report properties.

► Task 1: Start Report Manager

- Use Internet Explorer in Administrator mode to open Report Manager at <http://NY-SQL-01/reports>.



Note: To start Internet Explorer in Administrator mode, right-click it and then click **Run as administrator**.

► Task 2: Examine the Product Catalog report

- View the **Product Catalog 2008** report in the AdventureWorks Sample Reports 2008 folder.
- Note that there is a document map on the left of the report together with a number of catalog pictures on the right.
- Use the document map to view the product details for the **Mountain-100** mountain bike.
- Notice that you can navigate through the report by using:
 - The **Next Page** button
 - The **Current Page** box
 - The **Find Text** box

- Use the **Current Page** field to go to page 11 in the report.
 - Search for the word **saddle** in the report.
- **Task 3: View the Company Sales report**
- In the AdventureWorks Sample Reports 2008 folder, open the **Company Sales 2008** report.
 - Notice that you can expand the product categories and the years to drill-down and find more detailed information.
- **Task 4: Export the Company Sales report to Excel**
- Using the **Export** list, export the report to Excel.
 - View the report in Excel.
- **Task 5: Examine Sales Order report properties**
- Using the **Properties** tab, view the **General** properties of the report.
Note that it is possible to rename the report, and to add a description.
 - View the **Data Sources** properties.

Results: After this exercise, you should understand how to generate and view reports through the Report Manager Web site.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting

Review Questions

1. What is the breakdown of the Reporting Lifecycle?
2. What does Report Server use now instead of IIS?

MCT USE ONLY. STUDENT USE PROHIBITED

Common Issues related to Report Server

Identify the causes for the following common issues related to Report Server and fill in the troubleshooting tips. For answers, refer to relevant lessons in the module.

Issue	Troubleshooting tip
Login failed for user 'UserName'.	
A connection cannot be made. Ensure that the server is running.	
Connection error, where login failed due to unknown user name or bad password.	

MCT USE ONLY. STUDENT USE PROHIBITED

Module 2

Authoring Basic Reports

Contents:

Lesson 1: Creating a Basic Table Report	2-3
Lesson 2: Formatting Report Pages	2-11
Lesson 3: Calculating Values	2-16
Lab: Authoring Basic Reports	2-24

Module Overview

- Creating a Basic Table Report
- Formatting Report Pages
- Calculating Values

In this module you will learn the fundamentals of report authoring. Report authoring is the process of taking user specifications for the presentation of data, and translating them into an organized and useful format.

Report authoring includes configuring data sources and data sets, creating tabular reports, summarizing data, and applying basic formatting.

Once a report has been created, you or a user, can run the report multiple times and print, or otherwise distribute the report.

Lesson 1

Creating a Basic Table Report

- Report Definition Language
- Options for Authoring a Report
- Accessing Data
- Creating a Table
- Creating Groups

This lesson is designed to help you use Report Designer to create a basic table report based on the AdventureWorks2008 database. You can also use Report Builder or the Report Wizard to create reports. In this lesson, you will learn the fundamentals of creating a report project, set up connection information, define a query, add a Table data region, group and total some fields, and preview the report.

Report Definition Language

XML representation of the report

- **Open schema**

- Provides common standard for report definition
- Facilitates validation against schema
- Extensible

- **Output format neutral**

Key Points

A report definition contains data retrieval and layout information for a report. Report Definition Language (RDL) is an XML representation of this report definition. Because RDL defines a report independently of the authoring tool, you can create reports by using Report Designer or any third-party tool that supports the RDL schema. Report Designer automatically translates between the graphical representation of the report layout and RDL.

- The RDL for a particular report is validated by using an XML schema definition (XSD). The XSD defines the accepted elements of the XML document. When a report is published by using Report Designer, the XSD is used to validate the report.
- RDL is an open schema with which developers can extend RDL with additional attributes and elements.
- RDL is designed to be output-format neutral. This means that a report written in RDL can be rendered in any of several output formats including HTML, Microsoft® Office Excel® format, and others.

Options for Authoring a Report



Report Designer

Report development tool for developers



Report Builder

Simple web-based report builder



Model Designer

Creates the foundational data source and data sets for Report Builder to base reports upon

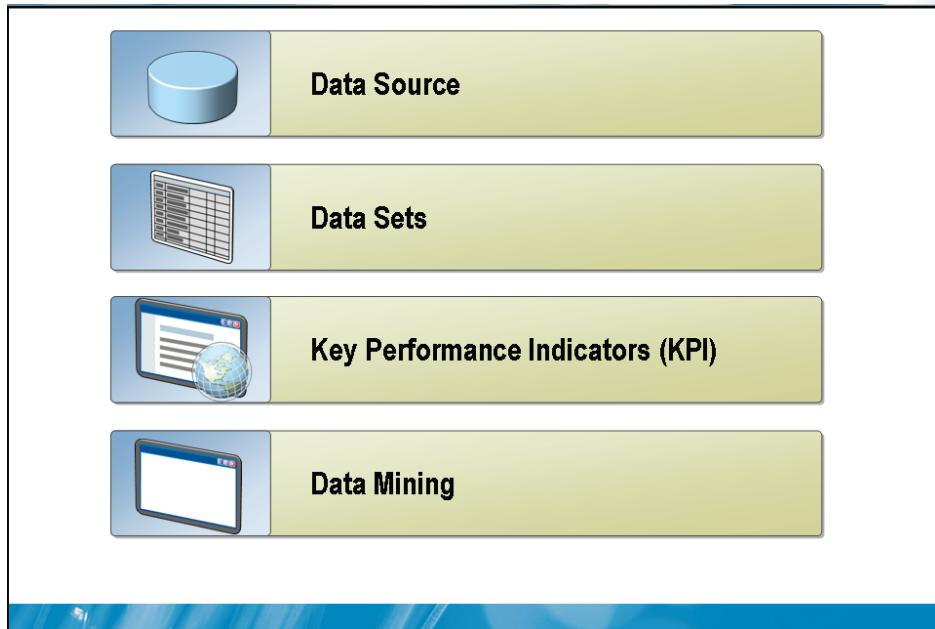
Key Points

Report Designer is the report creation tool that is primarily used by developers. This tool is available in SQL Server® Business Intelligence Development Studio(BIDS), and allows you to create a report from start to finish. This means that you use BIDS to create the data sources, data sets, and report layouts.

Report Builder is a report creation tool that is designed for the less technical business users rather than developers. Users access Report Builder from the Report Manager. The users can then create reports based on a report model, which is created by a developer to provide an abstraction layer between the Report Builder tool and the underlying data sources.

Model Designer is accessed through BIDS. Although Model Designer is not a report authoring tool, it is an important component in building reports using Report Builder. The models created by Model Designer contain all relevant information about how data is accessed, related, and presented to the business user. These models then become the foundation upon which the business user creates reports in Report Builder.

Accessing Data



Key Points

A Reporting Services data source contains information about a connection to a database. This includes such information as a server name, a database name, and user credentials. A Reporting Services data set contains information about the query to be used by a report. The fields contained in a data set are used as the basis for creating a report.

- A data source represents a connection to an external data source.
- A data source can be used to access data from any .NET Framework-managed data provider.
- Reporting Services directly supports Microsoft SQL Server 7.0 or later, OLE DB data sources, open database connectivity (ODBC) data sources, and Oracle.

- A report can retrieve values from multiple data sources.
- You can store user credentials for a data source, specify that the user should be prompted for credentials when the report executes, or use the user's Microsoft Windows® credentials when connecting to the data source.

Question: What types of non-native SQL Server data do you think you will be able to make reports from?

Creating a Table

Table is the simplest data region

- **Fields**
 - Display column values from data set
`=Fields!SalesAmount.Value`
- **Aggregate expressions**
 - Display summary values for groupings
`=Sum(Fields!SalesAmount.Value)`

Reseller [Country]	Sales Amount	Cumulative %	Order Quantity	Margin	Margin %
[State]	<code>=Sum(SalesAmount)</code>		<code>=Sum(OrderQuantity)</code>	<code>=Sum(Margin)</code>	<code>=Sum(Margin) / Sum(SalesAmount)</code>
[Reseller]	<code>=Sum(SalesAmount)</code>	<code>=Sum(Cumulative %)</code>	<code>=Sum(OrderQuantity)</code>	<code>=Sum(Margin)</code>	<code>=Sum(Margin) / Sum(SalesAmount)</code>
State Total	<code>=Sum(SalesAmount)</code>		<code>=Sum(OrderQuantity)</code>	<code>=Sum(Margin)</code>	<code>=Sum(Margin) / Sum(SalesAmount)</code>
Country Total	<code>=Sum(SalesAmount)</code>		<code>=Sum(OrderQuantity)</code>	<code>=Sum(Margin)</code>	<code>=Sum(Margin) / Sum(SalesAmount)</code>
Grand Total	<code>=Sum(SalesAmount)</code>		<code>=Sum(OrderQuantity)</code>	<code>=Sum(Margin)</code>	<code>=Sum(Margin) / Sum(SalesAmount)</code>

Key Points

Typically, the body of a report contains repeating data retrieved from an underlying data set. A data region is a control that is capable of both displaying data in a structured fashion and displaying database fields, custom fields, and expressions.

The simplest type of data region is a table. A table data region is made up of a group of cells organized into fixed columns with repeating rows. Each cell in a table contains one report item, such as a field or text box. The value displayed in the text box can be a string constant, a field from the current row of the data set, or a more complex expression.

- Using Report Designer, you can add fields to a table from the Datasets window or by typing an expression in to a text box.
- You can add aggregate expressions to create totals, averages, counts, maximum, and minimums values.

Creating Groups

Table data regions allow grouping of data

- **Table row types include:**
 - Detail row
 - Header/footer for grouping
 - Header/footer for the table

Reseller	Sales Amount	Cumulative %	Order Quantity	Margin	Margin %
[Country]					
[State]	=SalesAmount		=OrderQuantity	=SumMargin	=Expr
[Reseller]	=SalesAmount	=Expr	=OrderQuantity	=Margin	=Expr
State Total	=SalesAmount		=OrderQuantity	=SumMargin	=Expr
Country Total	=SalesAmount		=OrderQuantity	=SumMargin	=Expr
Grand Total	=SalesAmount		=OrderQuantity	=SumMargin	=Expr

Key Points

Table data regions provide a grouping mechanism that enables the display of detail rows, as well as header and footer rows for each grouping level contained within the report. Reports can, therefore, contain multiple levels of totaling.

- A group has a name and a set of group expressions that you specify.
- The set of group expressions can be a single dataset field reference or a combination of multiple expressions.
- In Report Designer, a group is a named set of data from the report dataset that is bound to a data region.
- After you create a group, you can set data region-specific properties, such as filter and sort expressions, page breaks, and group variables to hold scope-specific data.

Demonstration: Report Designer

In this demonstration, you will see how to:

- Launch Report Designer
- Create a Data Source
- Create a Data Set

Question: In this demo you were shown how to create a data source, a data set, and a basic report using the wizards available in Report Designer. All of these were created in a specific sequence. What importance do you see in the sequence that these objects must be created?

Lesson 2

Formatting Report Pages

- Report Page Options
- Report Items
- Headers and Footers

In this lesson you will learn the fundamentals of report formatting to allow for easier presentation and interpretation of data.

Formatting is an important factor in how your output will be displayed for your organization. The better your understanding of formatting, the easier it will be to create reports that are easy to read and understand.

Report Page Options

Customizable report page options

- Page headers and footers
- Page height and width
- Interactive features
- Page break options

Key Points

You can customize the way a report looks with a variety of different layout options, including report headers and footers, page height and width, and page breaks.

- A report can contain a header and footer that run along the top and bottom of each page. Headers and footers can contain text, images, and other report items, but they cannot contain data regions, subreports, or an item that refers directly to a field. In Report Designer, you can enable or disable page headers and footers by using the appropriate item within the **Report** menu.
- Page break options include: Logical page breaks, Hard Page-break renderers, and Soft Page-break renderers.
- You can format the report body so that there is a border color, border style, and border width. You can also add a background color and background image.

Report Items

Item	Types
Data region	<ul style="list-style-type: none">● List (Tablix)● Table (Tablix)● Matrix (Tablix)● Chart● Gauge
Independent items	<ul style="list-style-type: none">● Line● Text box● Image● Rectangle● Subreport

Key Points

Report items add data, structure, and formatting to a report and come in two varieties: data regions and independent items. You can add these to a report page by using the Toolbox in Report Designer.

Data region items render data from an underlying data set.

Data region items include the following:

- **List.** A list presents data arranged in a free-form fashion.
- **Table.** A table presents data in a row-by-row format.
- **Matrix.** A matrix, also known as a crosstab, contains both columns and rows that expand to accommodate data.
- **Chart.** A chart presents data graphically.
- **Gauge.** A gauge displays an indicator within a range of values.

Independent items add information and formatting to a report that is often not data related.

Independent items include the following:

- **Line.** Horizontal, vertical, and diagonal lines can be added to a report to provide emphasis or separation of data.
- **Textbox.** Textboxes allow you to enter additional information.
- **Image.** Images can be added to reports.
- **Rectangles.** Rectangles, like lines, can be added to provide emphasis to data. They can also contain other report items.
- **Subreport.** Subreports are additional reports that are included on a report to offer supporting data.

Question: One of the less common report items is a gauge. What types of reports do you use that a gauge or several gauges would enhance the information presented?

Headers and Footers

Level	Description
Report	<ul style="list-style-type: none">Area of report above or below data regionAppears only once for a report
Page	<ul style="list-style-type: none">Area at top or bottom of each pageOptional for first and last pages
Table	<ul style="list-style-type: none">Area above or below each tableHeader typically displays field captionsOption to repeat on each page
Group	<ul style="list-style-type: none">Area above or below each groupingForce page break before or afterOption to repeat on each page

Key Points

Headers and footers give context to a report. By using them, you can provide extra information to the report viewer, such as page numbers or the report print date.

- For reports with more than one dataset, you cannot add fields or data-bound images directly to a header or footer.
- You can use image data stored in a database in a header or footer. However, you cannot reference database fields from the Image report item directly.
- If you are only using the text box to populate a header or footer, you can hide the text box in the report body.
- In Office Excel, page footers have a limited layout. If you define a report that includes complex report items in the page footer, the page footer won't process as you expect when the report is viewed in Excel.

Question: What importance do headers and footers play in report design?

Lesson 3

Calculating Values

- Creating Custom Fields
- Aggregate Functions
- Common Aggregate Functions
- Collections
- Conditional Formatting

When designing a report, you are often confronted with situations that require different information to be displayed than what is stored in the database. This may include such information as calculated amounts, aggregates of data, or system information.

You can create custom fields with calculated values, and place them on your reports.

Creating Custom Fields

Calculated

- Based on database fields
- Based on system variables
- Display parameter values
- Can handle special values such as NULL
- Evaluated for every record returned in the data set

Key Points

When creating a data region within a report, you can use all of the fields in the underlying data set. You can also add custom fields by using Report Designer to provide calculated values based on data fields and global data held within the report.

- Calculated fields use expressions to derive new values for use in the report.
- Any database fields that are referenced in calculated fields must come from the same source data set. You can reference calculated fields in the same manner as database fields.
- Calculated fields can be used in the same manner as any other field available to the report.

Question: For what situations and reports do you see custom fields being of value to your organization?

Aggregate Functions

Syntax

- Function(Expression, Scope)

```
=Aggregate(Fields!LineTotal.Value,  
"GroupbyOrder")
```

- Expression – any valid expression, typically numeric
- Scope – grouping name specifying level for applying aggregate

Key Points

Aggregate functions enable you to return a single value from multiple rows in a data set, such as a total or an average.

In a Tablix data region, you can display aggregate totals for a report dataset, a data region, or a group. You can use the default aggregation supplied by the **Add Total** command and use the default scope. Alternatively, you can specify a different aggregate function from the built-in functions or specify a different scope.

The default scope can be a dataset, a data region, a Tablix group, the intersection of a Tablix row and column group, or a chart group. This provides flexibility and choice.

- Most aggregate functions follow a pattern similar to the following for the function arguments:

```
Function(Expression, Scope)
```

- **Expression** is typically a numeric field from the data set, but it can be any valid expression. The expression is calculated for each detail row accessed by the aggregate function.
- **Scope** determines which detail rows from the data set are accessed by the function. The aggregate applies only to rows that share a common value at the specified scope. The scope argument for an aggregate function allows two possibilities.

Common Aggregate Functions

Function	Description
Avg	Average of non-null values
Count	Count of values
CountDistinct	Count of all distinct values
CountRows	Count of rows within the specified scope
First	First value
Min	Minimum non-null value
RunningValue	Running aggregate (specify function)
StDev	Standard deviation of non-null values

Key Points

Reporting Services includes a small but useful set of aggregate functions. These functions allow you to encapsulate information from a large number of rows into a single, high-level value that can be displayed on a report.

- **Avg.** This function retrieves the sum of all values divided by the count of non-null values.
- **Count.** This function retrieves the count of all non-null values. This counts all the rows of the data source, not just distinct values.
- **CountRows.** This function returns a count of rows within the specified scope. Unlike **CountDistinct**, this function counts duplicates as individual values.
- **First.** This function retrieves the first value from a field over the scope. This is typically the value you get from the data set if you do not use an aggregate function, but explicitly using the **First** function makes the intent of the expression clearer.

Collections

Collection	Description
Fields	Fields from the dataset
Globals	Global variables such as ReportName
Parameters	Report parameters
ReportItems	The text boxes within the report
User	Data about the user running the report such as UserId
DataSources	Data sources referenced from within the report body
DataSets	Datasets referenced from the report definition
Variables	Collection of report variables and group variables

Key Points

There are several global collections that can be used to reference data values in a report. These global collections are often used within in custom fields on a report.

Each dataset in a report contains one Fields collection. The Fields collection is the set of fields specified by the dataset query, plus any additional calculated fields that you create.

Report parameters are one of the built-in collections you can reference from an expression. By including parameters in an expression, you can customize report data and appearance based on choices a user makes.

You can create a report variable or a group variable. A report variable is set once and can be used in expressions throughout a report.

The ReportItems collection includes text boxes that are in the current scope of a page header, page footer, or report body.

Conditional Formatting

- Use expressions to implement dynamic formatting
- Common uses
 - Make negative values red and positive values black
 - Create “greenbar” effect by coloring alternate rows
- Example expression

```
=IIF(Me.Value < 0, "Red", "Black")
```

Key Points

When formatting reports, it is sometimes useful to highlight, or hide, particular occurrences of data values or ranges of values. You can use conditional formatting to achieve this.

- Conditional formatting ties report formatting to the evaluation of an expression.
- Most appearance properties can include conditional formatting.
 - Dynamically setting the font or background color to highlight exceptions. For example, a report can draw attention to negative values by formatting them with a red font color or positive values by using a black font color.
 - Changing the background color of alternating rows to make a long report easier to read. Alternating row colors allow the report reader to visually align row values more easily.

For example, to conditionally format the color of a text box, you can set **Color** property to the following expression:

```
=IIF(Me.Value < 0, "Red", "Black")
```

Lab: Authoring Basic Reports

- Exercise 1: Creating a Basic Table Report
- Exercise 2: Formatting Report Pages
- Exercise 3: Adding Calculated Values to a Report

Logon information

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

Estimated time: 60 minutes

Exercise 1: Creating a Basic Table Report

Scenario

You will create a Product Profitability report by using the AdventureWorks2008 database. To do this, you will define a data source and data set, and then add groupings and basic formatting to a report table.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Create a Reporting Services Project.
3. Create a shared data source.
4. Create a report.
5. Add a dataset.
6. Add a table data region.
7. Add fields to the detail row.

8. Add fields to the table footer.
9. Create a Product Category group.
10. Create a Product SubCategory group.
11. Apply basic formatting to the table.
12. Indent row captions.

► **Task 1: Set up the lab environment**

- Start **6236A-NY-SQL-01** and log on as **Student** using the password **Pa\$\$w0rd**.
- Run the **runScripts.bat** batch file in the E:\MOD02\Labfiles\Starter folder.

► **Task 2: Create a Reporting Services Project**

- Using Business Intelligence Development Studio, create a new Report Server project named **Product Reports**.

► **Task 3: Create a shared data source**

- Add a shared data source to the report project and name it **AdventureWorksDW2008**.
- Use the localhost server.
- Use Windows Authentication.
- Use the AdventureWorksDW2008 database.
- Test the connection before proceeding to the next task.

► **Task 4: Create a report**

- Add a new item to the project. This item should be a report named **Product Profitability**.
- Do not use the Report Wizard.

► **Task 5: Add a dataset**

- Create a dataset named **DataDetail** that uses the AdventureWorksDW2008 shared data source.
- Use the following query string:

```
SELECT * FROM vProductProfitability WHERE Year = 2003 AND  
MonthNumberOfYear = 1
```

► **Task 6: Add a table data region**

- Add a table data region to the report.

► **Task 7: Add fields to the detail row**

- Using the Report Data pane, add the following fields to the detail row of the table, adjusting the column widths to be sure that all values are visible. You may have to click to the Preview tab to verify that the width is sufficient.
 - **Product**
 - **SalesAmount**
 - **OrderQuantity**

► **Task 8: Add fields to the table footer**

- In the **Table Footer**, add a **Grand Total** caption to the first cell, and add summary totals for the **SalesAmount** and **OrderQuantity** fields.
- Preview the report. Make any necessary adjustments before continuing.

► **Task 9: Create a Product Category group**

- Add a group named **Category** to the detail row, grouping on the following expression:

```
=Fields!Category.Value
```

- Add the **Category** field found in the **DataDetail** dataset to the first cell of the **Category Header** row.
- In the **Category footer** row, add a **Category Total** caption to the first cell.
- Copy the **SalesAmount** and **OrderQuantity** expressions from the **Table Footer** and paste them into the **Category Footer**.

► **Task 10: Create a Product SubCategory group**

- Add a group to the detail row, grouping and sorting on the following expression:

```
=Fields!SubCategory.Value
```

- Add the **SubCategory** field found in the **DataDetail** dataset to the first cell of the **SubCategory Header** row.
- In the **Category Footer** row, add a **SubCategory Total** caption to the first cell.
- Copy the **SalesAmount** and **OrderQuantity** expressions from the **Table Footer** and paste them into the **SubCategory Footer**.

► **Task 11: Apply basic formatting to the table**

- Add a group to the detail row, grouping and sorting on the following expression:

```
=Fields!SubCategory.Value
```

- Format the **Sales Amount** column as **currency** with no decimal places (C0).
- Format the **Order Quantity** column as **numeric** with no decimal places (N0).
- Set the background color of the detail row to **WhiteSmoke**.
- Format the **Table Header** and **Footer** as follows:
 - **12pt font, bold.**
 - **White background.**
 - **Black foreground.**
- Format the **Category Header** and **Footer** as follows:
 - **12pt font, bold.**
 - **Background** set to **Gainsboro**.
- Format the **SubCategory Header** and **Footer** to use **11pt font, bold**.

► **Task 12: Indent row captions**

- Set the left padding for the Detail, SubCategory Header, and Footer caption cells to **12pt**.
- Preview the report. Make any necessary adjustments before finishing.

Results: After this exercise, you have created a report, added report data, and then added grouping and formatting to the report.

Exercise 2: Formatting Report Pages

Scenario

You will apply additional formatting to the report that you created by adding headers and footers, images, and page breaks.

The main tasks for this exercise are as follows:

1. Open the Product Reports solution.
2. Add a page break at the end of each category group.
3. Add a report header to the first page of the report.
4. Add an image to the report.
5. Add a repeating table header.
6. Add a page header to the report.
7. Add a page footer to the report.

► Task 1: Open the Product Reports solution

- Using Business Intelligence Development Studio, open the **Product Reports** solution located in the E:\MOD02\Labfiles\Starter\Product Reports folder.



Note: You can omit this step if you already have the Product Reports project open from Exercise 1.

► Task 2: Add a page break at the end of each category group

- Open the **Product Profitability** report.
- Edit the **Category Footer** group so that a page break is inserted at the end of each Category.

► **Task 3: Add a report header to the first page of the report**

- Move the table item down the page so that there is approximately a **1.5 inch (4 cm)** gap at the top of the report.
- Add a **12pt** horizontal line to the top of the report.
- Add a textbox to the top left corner.
 - Adjust the width to **2.5 inches (6 cm)** wide.
 - Adjust the height to **1.25 inches (4 cm)** high.
- Add the report title: **Adventure Works Product Profitability Report**.
- Set the font size to **20pt, bold**.

► **Task 4: Add an image to the report**

- Add an image to the right of the report title.
- Use the **logopart.jpg** image located in the E:\MOD02\Labfiles\Starter folder.

► **Task 5: Add a repeating table header**

- Alter the Table Header, setting the **RepeatOnNewPage** property to **True**.
- Preview the report. Check that the product header appears on each page.

► **Task 6: Add a page header to the report**

- Add a page header to the report.
- Add a textbox to the page header.
 - Adjust the width to **2 inches (5 cm)** wide.
 - Adjust the height to **0.25 inches (0.5 cm)** high.
 - Move the textbox to the right edge of the report.
 - Add the text: **Product Profitability Report**.

- Set the textbox alignment to **Right**.
- Alter the following properties of the page header.
 - Set **PrintOnFirstPage** to **False**.
 - Set **PrintOnLastPage** to **True**.

► **Task 7: Add a page footer to the report**

- Add a page footer to the report.
- Add a textbox to the page footer.
 - Adjust the width to **1.5 inches (4 cm)** wide.
 - Adjust the height to **0.25 inches (0.5 cm)** high.
- Using an italic font, add the text: **Company Confidential**.
- Horizontally center the textbox relative to the report page.
- Ensure that the following page footer properties are set.
 - **PrintOnFirstPage: True**
 - **PrintOnLastPage: True**
- Preview the report. Check the following:
 - Each category is on a different page.
 - Table header and page footer are repeated on all pages of the report.
 - The page header appears on all pages except the first page.

Results: After this exercise, you have customized report formatting by adding page breaks, headers and footers, and an image.

Exercise 3: Adding Calculated Values

Scenario

You will add calculated values to your report and use conditional formatting to create summary totals for all grouped information.

The main tasks for this exercise are as follows:

1. Open the Product Reports solution.
2. Create and add a calculated field for the profit margin.
3. Add the Margin calculated field to the report.
4. Add a margin percentage expression to the detail row.
5. Add a margin percentage expression to the table footer.
6. Add a margin percentage expression to the Category footer.
7. Add a margin percentage expression to the SubCategory footer.
8. Conditionally format rows with low margin percents.
9. Copy conditional formatting to additional rows.
10. Sort products in descending order of Sales Amount.
11. Show cumulative Sales Amount within SubCategory.
12. Convert the Cumulative value to a percentage.

► Task 1: Open the Product Reports solution

- Using Business Intelligence Development Studio open the **Product Reports solution** located in the E:\MOD02\Labfiles\Starter\Product Reports folder.



Note: You can omit this step if you already have the Product Reports project open from Exercise 1.

► **Task 2: Create and add a calculated field for the profit margin**

- Open the Product Profitability report.
- Using the **Datasets** window, add a new field to the **DataDetail** dataset.
 - Set the name to **Margin**.
 - Select **Calculated Field**.
 - Add the following expression:

```
=Fields!SalesAmount.Value - Fields!CostAmount.Value
```

► **Task 3: Add the Margin calculated field to the report**

- Insert a column to the right of the **Order Quantity** column.
- Using the new column:
 - Add the **Margin field** to the detail row.
 - Add the **Margin field** to the **SubCategory Footer** row.
- Copy the expression from the **SubCategory Footer** row and paste it into the **Category Footer** row, and the **Table Footer** row.
- Set the **Format** property of the column to **C0**.
- Preview the report. Make any necessary adjustments before continuing.

► **Task 4: Add a margin percentage expression to the detail row**

- Insert a column to the right of the **Margin** column.
- Using the new column:
 - In the **Table Header** row, type **Margin %**.
 - Set the **Format** property of the column to **P1**.
 - Type the following expression in the **Detail** row:

```
=Fields!Margin.Value / Fields!SalesAmount.Value
```

- Preview the report. Make any necessary adjustments before continuing.

► **Task 5: Add a margin percentage expression to the table footer**

- Using the **Table Footer** row.
 - In the **Sales Amount** cell, set the **Name** property to **SalesAmount_Total**.
 - In the **Margin** cell, set the **Name** property to **Margin_Total**.
 - In the **Margin %** cell, add the following expression:

```
=ReportItems!Margin_Total.Value /  
ReportItems!SalesAmount_Total.Value
```

- Preview the report. To view the **Margin %** Table Footer, navigate to the end of the report.

► **Task 6: Add a margin percentage expression to the Category footer**

- Using the **Category Footer** row.
 - In the **Sales Amount** cell, set the **Name** property to **SalesAmount_Category**.
 - In the **Margin** cell, set the **Name** property to **Margin_Category**.
 - In the **Margin %** cell, add the following expression:

```
=ReportItems!Margin_Category.Value /  
ReportItems!SalesAmount_Category.Value
```

- Preview the report. Notice that a margin percent exists for each category.

► **Task 7: Add a margin percentage expression to the SubCategory footer**

- Using the **SubCategory** Footer row.
 - In the **Sales Amount** cell, set the **Name** property to **SalesAmount_SubCategory**.
 - In the **Margin** cell, set the **Name** property to **Margin_SubCategory**.
 - In the **Margin %** cell, add the following expression:

```
=ReportItems!Margin_SubCategory.Value /  
ReportItems!SalesAmount_SubCategory.Value
```

- Preview the report. Notice that a value for margin percent now exists at subcategory level.

► **Task 8: Conditionally format rows with low margin percents**

- Add the following expression to the **Detail** row of the **Margin %** column:

```
=IIF(Me.Value<0.15, "Red", "Black")
```

► **Task 9: Copy conditional formatting to additional rows**

- Using the **Margin %** column.
 - Use the **Properties** window to copy the expression in the **Color** property of the **Detail** row.
 - Paste the expression into the **Category Footer** and **Table Footer** rows.
 - Alter the expression in the **Table Footer** row, replacing "Black" with "White".

► **Task 10: Sort products in descending order of Sales Amount**

- Add a group to the **Detail** row.
- Name the group **DetailSort**.
- Group on the following expression:

```
=Fields!Product.Value
```

- Sort on the following expression:

```
=Fields!SalesAmount.Value
```

- Set the sort to **A-Z**.
- Preview the report. Notice that the products are listed in descending order based on sales amount.

► **Task 11: Show cumulative Sales Amount within SubCategory**

- Insert a column to the right of the **Sales Amount** column.
- Using the new column:
 - In the **Table Header** row, type: **Cumulative**
 - Type the following expression in the **Detail row**:

```
=RunningValue(Fields!SalesAmount.Value, Sum, "SubCategory")
```

- Preview the report. Notice the Cumulative value increases within each subcategory until it matches the total for the subcategory.

► **Task 12: Convert the Cumulative value to a percentage**

- Using the **Cumulative** column:
 - Edit the expression in the **Detail** row as follows:

```
=RunningValue(Fields!SalesAmount.Value, Sum, "SubCategory") / ReportItems!SalesAmount_SubCategory.Value
```
 - Set the **Format** property of the column to **P1**.
 - Change the label in the **Table Header** row to **Cumulative %**.
- Preview the report and make any adjustments as necessary. Notice that the cumulative percentages accumulate to 100% for each subcategory and then restart with each new subcategory.
- Turn off virtual machine and discard changes.

Results: After this exercise, you have added calculated fields, conditional formatting, and sorting to a report.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips

Review Questions

1. Grouping can be done by rows or columns. You have seen examples of grouping by rows in this module. What kinds of reports do you think are more suitable for grouping by columns and why?
2. Conditional formatting is an important concept in report design. Other than changing colors, what kinds of conditional formatting do you think you will use in your reports?

Common Issues related to Report Server

Identify the causes for the following common issues related to Report Server and fill in the troubleshooting tips. For answers, refer to relevant lessons in the module.

Issue	Troubleshooting tip
A network error occurs when launching Report Manager as http://localhost/reports	
Exporting Reports that require a missing client component	
Selected Pages in Report Manager do not open	

MCT USE ONLY. STUDENT USE PROHIBITED

Module 3

Enhancing Basic Reports

Contents:

Lesson 1: Interactive Navigation	3-3
Lesson 2: Displaying Data	3-11
Lab: Enhancing a Report	3-24

Module Overview

- Interactive Navigation
- Displaying Data

You will learn about how reports can be enhanced by using features such as navigation, and different data regions such as charts and subreports. Navigation helps users to interactively view reports in a more usable way, and data regions allow you to display data in different formats to suit the needs of the report.

Lesson 1

Interactive Navigation

- Using Dynamic Visibility
- Creating a Document Map
- Initiating Actions

In environments that support interactivity, such as HTML 4.0, reports can include a variety of features that provide interactivity to users. You can enable a user to control report display and content by designing expressions that include parameter references for sort, filter, and visibility. You can use a document map in a report to provide users with a way to navigate to certain areas of the report. Use drillthrough reports to enable a user to view a main report and drill through to details or other related data by clicking a link. You can add interactive sort buttons to enable a user to toggle between ascending and descending order for rows in a table or for rows and columns in a matrix.

Using Dynamic Visibility

Use show/hide to provide drill-down interactivity

- Allows users to navigate from summarized data to detailed data
- Implemented by adding the visibility toggle to the detail, row group, or column group
- Only supported by rendering extensions that support interactivity, such as HTML

Key Points

By using dynamic visibility, you can create a report that enables the user to show or hide the detail it contains. You can hide any report item based on the contents of other report items including groups, columns, or rows in a table or matrix.

- Hidden report items enable the user to toggle between summary and detail views (“drilling down”). When the report is rendered, the user can click the text box to expand and collapse the detail data.
- Follow these steps in Microsoft® SQL Server® 2008 Report Designer to create this drill-down effect:
 1. Select the group, column, or row to hide.
 2. Set its **hidden** state to **True**.
 3. Set the toggle item to the name of a text box in a containing group.

Depending on the rendering format chosen, hidden items may or may not be shown in the report. For example, when rendering reports, Hypertext Markup Language (HTML) will render show-and-hide toggles on report items and not render hidden items, whereas Extensible Markup Language (XML) will render all report items, regardless of whether they are hidden.

Creating a Document Map

Provides report navigation support

- Requires rendering mechanism that can support document maps
 - HTML, intended as primary
 - Microsoft Excel
 - Microsoft Word
 - PDF
 - TIFF, XML, and CSV ignore document maps

Key Points

Reports can contain large amounts of data that can be difficult to display effectively to a user. Users will not use a report if they find it too difficult to work with. To improve this situation, you can create a document map to improve navigability.

A document map appears as a table of contents for the report and provides links to particular areas of the report. To create a document map, add a **Document Map** label to a grouping level. When you render the report, the document map is automatically generated.

- When you view an HTML, Microsoft® Office Excel®, or Portable Document Format (PDF) report, a document map is displayed along the side of the report. Clicking an item in the document map refreshes the report and displays the area of the report that corresponds to the clicked item. Document maps are rendered for different formats as follows:
 - **HTML:** The document map appears as a panel on the left side of the document.

- **Excel:** The document map appears as a worksheet that contains a navigable tree structure, which hyperlinks to the actual data.
- **PDF:** The document map is generated as a set of bookmarks.
- All report items contain a **Document Map** label property. If any report items have a **Document Map** label, a document map is automatically generated when a user views the report in HTML Viewer or Adobe Acrobat Reader. Data regions and independent report items can appear in a document map.

Initiating Actions

Hyperlink	Link to a Web resource from a static URL or an expression
Drill-through report link	Navigate from report to report using links with option to pass parameters
Bookmark link	Navigate to another part of the same report

Key Points

You can add links to a report to help users navigate to particular areas of a report, to other resources located on the report server, or to resources located elsewhere that are identified by a valid URL.

- Hyperlinks in text boxes and images enable the user to jump to a particular URL. A hyperlink can link to a static URL resource or an expression that evaluates to a URL. To create dynamic hyperlinks, URLs can be stored in a database and rendered in a data region.

- Drill-through report links allow you to link reports together. Where a parameter is specified, the target report may be filtered using the passed parameter. Drill-through reports commonly contain details about an item that is contained in an original, separate summary report. Any report that is stored on the report server can be a drill-through report. You can only add drill-through links to text boxes and images.
- A bookmark link is a link that a user clicks to move to another area or page in a report. As with all interactive features, the behavior of links is dependent on the rendering engine.

Demonstration: Preview a Document Map

In this demonstration, you will see how to:

- Open a sample document map

Question: What grouping might you add to a document map in your organization?

Lesson 2

Displaying Data

- Data Regions
- Tables
- Charts
- Lists
- Matrices
- Gauge
- Tablix: Mixing Tables and Matrices
- Working with Data Regions
- Subreports

In addition to the general rendering behaviors that apply to all report items, data regions have additional pagination and rendering behaviors that they follow. Data region-specific rendering rules include how a data region grows, how special cells such as the corner cell or header cells are rendered, and how a data region for right-to-left reading is rendered. This lesson discusses how the various parts of a data region are rendered.

Data Regions

Data Region	Container Type	Grouping
Report body (or rectangle)	Free-form	Non-repeating
Table	Fixed	Multiple grouping levels on rows axis
Chart	N/A	Single grouping level, dynamically created
Lists	Free-form	Single grouping level, detail or grouped
Gauge	Fixed	Single grouping level
Tablix	Free-form	Multiple grouping levels on rows and columns
Matrix	Fixed	Multiple grouping levels on rows and columns

Key Points

A data region is a report item that displays repeating data. The choice of data region type is dependent on the structure of the underlying data set and the way in which data is to be displayed in the report.

- The report body or rectangle allows you to add other data regions or items such as text boxes in a free-form style. It does not support grouping directly, but contained data regions can support grouping.
- List data regions are a free-form container data region, which means that they can contain multiple items, freely arranged. A list has a single grouping level, but you can nest a list within a list to achieve multiple grouping levels.
- Table and Matrix data regions are fixed container data regions, which mean that a single cell of a table or matrix can contain only a single report item. A table can have multiple grouping levels, but only on the rows axis. A matrix can have multiple grouping levels on both the rows and columns axis.

- Chart data regions graphically represent data series. These data regions are not containers and therefore do not contain report items. A chart has a single grouping level that is dynamically created.
- Gauge presents data as a range with an indicator pointing to a specific value within the range. Gauges are used to display key performance indicators (KPIs) and other metrics. Examples of gauges include linear and circular.

Tables

Fixed tabular structure with one report item per cell

- **Multiple grouping levels**
 - Variable data in rows
 - Fixed columns
- **Formatting columns**
 - Padding
 - Merge cells
 - Rectangle in cell

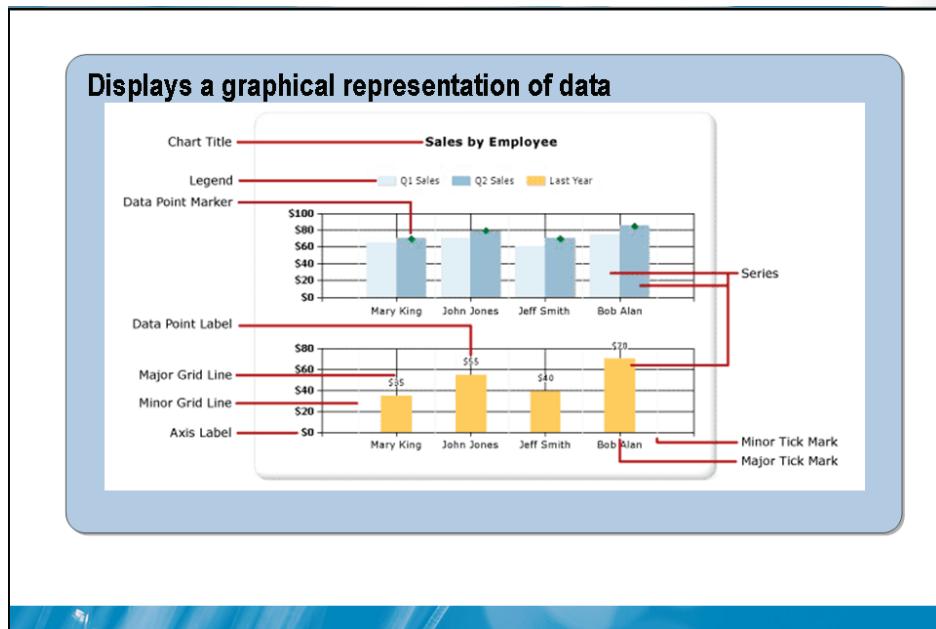
	Date	Order	Product	Qty	Line Total
≡	[Date]	[Order]	[Product]	[Qty]	[Line Total]

Key Points

A table is an effective way to display data that has a fixed set of columns with recurring detail rows. The Table data region contains other report items in a fixed manner—that is, each cell of the table can contain only one report item. This makes it easy to align columns of numbers and labels. Each cell in a table contains a text box by default. You can type any expression into any cell, or you can change the report item within the cell to another item.

- The Table data region allows you to add multiple levels of grouping, but for rows only. The definition of values on rows changes based on the current contents of the data set. Columns are fixed.
- You can use the following column formatting techniques to alter the appearance of tables:
 - **Padding** Adjusts the space between the edge of a cell and its contents.
 - **Merge cells** Combines two or more adjacent cells into one cell.

Charts



Key Points

The Chart data region displays a graphical representation of data in a report. You can add different types of charts to a data region, and you can specify values as well as category and series groups. Charts can include different colors, symbols, and three-dimensional (3D) effects.

- A chart must include at least one data value series. Values determine the size of the chart element for each category group.
- Each chart type has unique characteristics to help you visualize your dataset. You can use any chart type to display your data, but your data will be easier to read when you use a chart type that is suitable to your data, based on what you are trying to show in your report.
- To display a chart element for each value series, define multiple values. If there are multiple value series, the chart legend displays the name of each value series.

- After you add a Chart data region to the design surface, you can drag report dataset fields for numeric and non-numeric data to the drop zones of the chart.
- Category groups are used to group data. Categories provide the labels for chart elements. For example, in a column chart (described below), a category label is displayed on the X-axis for each set of columns.

You can nest category groups. When you define multiple category groups, each category is nested within another category.

Lists

A free-form container for complex repeating areas

- Repeating regions with free-form layout
- Single grouping level
- Nest lists to get multiple levels of grouping

The diagram illustrates a List data region. On the left, a design view shows a list structure with items: [Date], [Qty], [Product], and [LineTotal]. On the right, two instances of the list are shown. The first instance for July 01, 2001, contains an item for "Long-Sleeve Logo Jersey, L" with a blue shirt icon and ID S043677. The second instance for July 01, 2001, contains an item for "Road-150 Red, 56" with a red bike icon and ID S043677.

Key Points

Table data regions do not always provide the necessary functionality for your report design because they always appear in a tabular format. However, creating a List data region provides an alternative that is not fixed in its format and is therefore suitable for creating invoices or purchase orders.

- A List data region is a free-form container that permits the placement of report items anywhere within the list. The list repeats for each group or row in the underlying data set. A list can be used for free-form reports or in conjunction with other data regions.
- The List data region provides more flexibility than other kinds of data regions, such as the Table data region. However, the report author has to manually position each report item required in the list, and lists therefore take more effort to format.

- In a list, you can display detail data with one instance of the list rectangle for each row of the detail data set, or a grouping level with one instance of the list rectangle for each unique combination of values for the grouping. Unlike a table, a list allows only a single grouping. If you want to display multiple grouping levels, you must nest one list inside another.

Matrices

Also known as pivot table or crosstab reports

- Fixed container
 - Each cell contains one report item
- Multiple grouping levels (both rows and columns)
 - Cells repeat both horizontally and vertically
 - Supports dynamic rows and columns

		North America			Europe			Pacific	Total
		CA	US	DE	FR	GB	AU		
Clothing	Caps	\$4893.79	\$17991.48	\$1368.42	\$1775.84	\$3823.84	\$780.02	\$30633.39	
	Tights	\$23716.43	\$128981.51	\$1316.07	\$11746.67	\$31955.36		\$197716.05	
	Total	\$234,999.57	\$1,016,999.68	\$71,183.38	\$125,402.87	\$246,448.43	\$33,683.09	\$1,728,717.03	
Components	Chains	\$793.83	\$4972.00	\$449.33	\$1195.22	\$716.50	\$704.35	\$8831.23	
	Cranksets	\$18515.99	\$105118.64	\$9849.31	\$24542.22	\$22994.19	\$14158.36	\$195178.70	
	Touring Frames	\$196141.85	\$7671130.33	\$192614.94	\$173594.09	\$142874.93	\$111981.32	\$1584337.46	
	Total	\$1,274,168.00	\$7,327,076.03	\$314,268.87	\$860,159.50	\$1,571,664.98	\$183,141.55	\$11,530,478.93	
Total	\$9,535,865.57	\$52,692,325.91	\$1,827,066.71	\$4,509,888.93	\$8,503,338.65	\$1,421,810.92	\$78,490,296.69		

Key Points

The Matrix data region displays data as a series of columns and rows. As opposed to tables, columns in a matrix can be dynamic. Using Report Designer, you can define matrices that contain static and dynamic rows and columns.

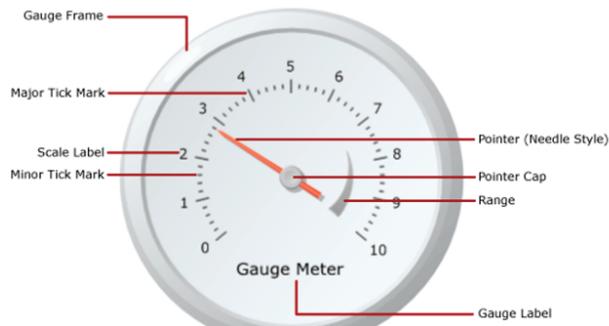
Matrices are also known as pivot tables or crosstabs. An example of a matrix report is a sales report that shows sale totals for a list of countries in specific years.

Because the list of years could change, a standard table layout would not provide the dynamic creation of columns that a matrix provides.

- In addition to creating automatic rows and columns based on data set values, you can also add static rows or columns using the Matrix data region. Static columns are similar to the fixed columns of a table report. Static rows or columns make it possible to add more than one aggregation for a single numeric field.
- In a matrix, there is no detail level. Even at the lowest level, you create a grouping level and use an aggregate function. The aggregate function you specify for the most detailed data is automatically used for all higher levels.

Gauge

Basic elements of a single gauge



Key Points

The Gauge data region is a one-dimensional data region that displays a single value in your dataset. An individual gauge is always positioned inside a gauge panel, where you can add child or adjacent gauges. You can use the gauge panel to create multiple gauges inside a single gauge panel that share common functions such as filtering, grouping, or sorting.

- Display key performance indicators (KPIs) in a single radial or linear gauge.
- Place a gauge inside a table or matrix to illustrate values inside each cell.
- Use multiple gauges in a single gauge panel to compare data between fields.

Tablix: Mixing Tables and Matrices

Matrix plus:

- Multiple parallel row/column members at each level
- Each member may be dynamic or static
- Optional omission of member headers

Table plus:

- Dynamic, nested column groups
- Multiple parallel row groups
- Static rows
- Optional spanning row headers

Key Points

The Tablix data region is a generalized layout report item that displays report data in cells that are organized into rows and columns. Report data can be detail data as it is retrieved from the data source, or aggregated detail data organized into groups that you specify. Each Tablix cell can contain any report item, including a text box, an image, or another data region such as a Tablix region, chart, or gauge.

- By default, a matrix has row groups and column groups and no detail group.
- By default, each Tablix cell in a table or matrix contains a text box. The cell in a list contains a rectangle.
- A Tablix data region has possible four areas for cells: the Tablix corner, the Tablix row group hierarchy, the Tablix column group hierarchy, and the Tablix body.

Working with Data Regions

Repeating data regions	Display of the same data region multiple times in a report
Empty data regions	Empty data displays the NoRows property value
Rendering dependencies	Use rectangles to fix the position of a data region so it does not expand
Sorting	Set default sort and/or allow user sort

Key Points

When you work with data regions, you should consider the nesting, rendering, and sorting of data regions in a final report.

- You can nest data regions to allow the display of the same data region multiple times in a report.
- A data region is only rendered if it contains data. If no data is returned, a text box containing the value of the **NoRows** property is displayed. **NoRows** can be set by using Report Designer.
- In some rendering formats, such as HTML, the position of other report items can change to accommodate expansion of a data region. Rectangles can be used to fix the position of a data region.

Subreports

A report inside the body of the main report

- Stored on the report server as another report
- Can be repeated within a data region in main report
- Only the body of the subreport is included
- Data source can differ from data source in main report
- Data regions often perform better than subreports

Key Points

Sometimes the detail of a report's data region is so complex that a separate report is a better option than using the previously described data region types. This is known as a subreport.

- A subreport can be repeated within data regions, using a parameter to filter data in each instance of the subreport.
- The subreport data source can be different from that of the parent report.
- Reports with data regions often perform better than reports that include subreports. When you run a report that contains a subreport, the report server has to process both reports.
- Display a standalone report inside another report.

Lab: Enhancing a Report

- Exercise 1: Using Dynamic Visibility
- Exercise 2: Using Document Maps
- Exercise 3: Initiating Actions
- Exercise 4: Using a List Data Region

Logon information

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

Estimated time: 60 minutes

Exercise 1: Using Dynamic Visibility

Scenario

The Adventure Works team has asked you to enhance a number of existing reports. In this lab you will need to enhance existing reports using navigational controls and data regions. You will also need to create a tablix report.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the AdventureWorks solution.
3. Preview the report.
4. Hide the Reseller detail and State footer rows.
5. Toggle the visibility of the hidden rows.
6. Show State totals when the detail is hidden.

- ▶ **Task 1: Set up the lab environment**
 - Start 6236A-NY-SQL-01 and log on as **Student** using the password **Pa\$\$w0rd**.

- ▶ **Task 2: Open the AdventureWorks solution**
 - Execute the **runScripts.bat** batch file in the E:\MOD03\Labfiles\Starter\Exercise01 folder.
 - Using Business Intelligence Development Studio, open the **AdventureWorks.sln** solution located in the E:\MOD03\Labfiles\Starter folder.



Note: You will need to run the Business Intelligence Development Studio as Administrator.

- ▶ **Task 3: Preview the report**
 - Using the **Preview** tab, view the **Reseller Sales** report.
 - Notice that the report contains detail and summarized values for resellers, subcategories, and categories, including both header and footer summaries for subcategories and categories.
- ▶ **Task 4: Hide the Reseller detail and State footer rows**
 - Using the Properties window, alter the **Visibility** property for the **Detail** row (this row has [Reseller] in the left column and is the fifth row from the top).
 - Set the **Hidden** property to **True**.
 - Using the Properties window, alter the **Visibility** properties for the **State Total** footer row.
 - Set the **Hidden** property to **True**.
 - Preview the report.
 - The detailed reseller rows and state footers are now hidden.

► **Task 5: Toggle the visibility of the hidden rows**

- In the Properties window, alter the **Visibility** property for the **Detail** row (this row has [Reseller] in the left column and is the fifth row from the top).
 - Set the **ToggleItem** property to **State**.
- In the Properties window, alter the **Visibility** property for the **State Total** footer row.
 - Set the **ToggleItem** property to **State**.
- Preview the report.
 - Expand Connecticut to view the resellers for this state. Notice that the plus sign allows you to view the detailed reseller and state total footer rows.

► **Task 6: Show State totals when the detail is hidden**

- In the **State Total** Footer row, copy the following cells:
 - **Sales Amount**
 - **Cumulative %**
 - **Order Quantity**
 - **Margin**
- Paste the cells into **Sales Amount** cell in the **State Header** row.
- In the Properties window, alter the **Visibility** property for the selected rows.
 - Set the **ToggleItem** property to **State**.
- Preview the report.
- Expand Texas to view the resellers for this state. Notice that the data next to Texas in the state header row is now hidden when you drill down.

Results: After this exercise, you should have successfully previewed the report, hidden rows, toggled visibility, and preview the report after making changes.

Exercise 2: Using Document Maps

Scenario

The Product Catalog report contains a detailed listing of more than 350 Adventure Works products. In this exercise, you will create a document map to provide a navigation mechanism for users of this lengthy report.

The main tasks for this exercise are as follows:

1. Open the Document Map solution.
2. Preview the report.
3. Add a label to the group in SubCategoryList.
4. Add a label to the group in ModelList.
5. Save and preview the report.

► **Task 1: Open the Document Map solution**

- Using Business Intelligence Development Studio open the **Document Map.sln** solution located in the E:\MOD03\Labfiles\Starter\Exercise02 folder.
- Open the **Product Catalog** report.

► **Task 2: Preview the report**

- Preview the **Product Catalog** report.
- Use the document map to navigate to the clothing products, and view the product details shown in the catalog.

► **Task 3: Add SubCategoryList to the document map**

- Using the Properties window view the properties of the **SubCategoryList**.
- Edit the **Row Group** property, setting the **Document map** label to **ProdSubCat**.

► **Task 4: Add ModelList in the document map**

- Using the Properties window view the properties of the **ModelList**.

- Edit the **Row Group** property, setting the **Document map** label to **ProdModel**.
- **Task 5: Save and preview the report**
- Save and then preview the report.
 - Use the document map to navigate to the Clothing product category. Notice that the products are now grouped by Product, SubCategory and model.

Results: After this exercise, you should have successfully previewed the report, added a SubCategoryList to the document map, made changes to the document map, and previewed the report after making the changes.

Exercise 3: Initiating Actions

Scenario

You have been asked to create an Order Details report. Adventure Works uses this report to verify the status of orders, to troubleshoot order complaints, and to answer questions from resellers. To troubleshoot reseller inquiries, they would like to be able to jump to the product detail information. In this exercise, you will add an action in the Order Details report to jump to the Product Detail report.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the Action solution.
3. Preview the report.
4. Add an action to jump to the report.
5. Save and preview the report.

► **Task 1: Set up the lab environment**

- Execute the **runScripts.bat** batch file in the E:\MOD03\Labfiles\Starter\Exercise03 folder.

► **Task 2: Open the Action solution**

- Using Business Intelligence Development Studio open the **Action.sln** solution located in the E:\MOD03\Labfiles\Starter\Exercise03 folder.

► **Task 3: Preview the report**

- Open the **Product Details** report.
- Preview the report and then save it.
- Open the **Order Details** report.
- Preview the report. Notice that the report contains a series of orders for a single customer.

► **Task 4: Add an action to jump to report**

- Add an action to the **Order** column in the **Detail** row (this row has **[Product]** in the left column and is the third row from the top).
 - Configure the action to jump to the **Product Detail** report.
 - Add a parameter named **Product**.
 - Set the value of the parameter to **Product**.
- Set the **TextDecoration** of the **Order** cell to **Underline**.

► **Task 5: Save and preview the report**

- Save and then preview the report.
- Click the product named **Road-650 Red, 60** in the report to jump to the **Product Detail** report. Notice that the report is filtered by product number.

Results: After this exercise, you have successfully previewed the report, added an action to the report, and saved and previewed the report after making changes.

Exercise 4: Using a List Data Region

Scenario

In this exercise, you have been asked to create a basic product sales report by using the List data region. Lists require more development effort from the report author, but provide a greater level of flexibility than tables.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the Data Regions solution.
3. Preview the report.
4. Copy the report.
5. Expand the height of the report.
6. Add a Category list.
7. Add fields to the Category list.
8. Add header and footer rows.
9. Add SubCategory list.
10. Add fields to the SubCategory list.
11. Combine the Category and SubCategory lists.
12. Delete the table version of the report.

► **Task 1: Set up the lab environment**

- Execute the **runScripts.bat** batch file in the E:\MOD03\Labfiles\Starter\Exercise04 folder.

► **Task 2: Open the Data Regions solution**

- Using Business Intelligence Development Studio open the **Data Regions** solution located in the E:\MOD03\Labfiles\Starter\Exercise04 folder.

► **Task 3: Preview the report**

- Preview the **Table Product Profitability** report. Notice that it is a tabular report, containing a header and footer together with products grouped by Product Category and SubCategory.

► **Task 4: Copy the report**

- Create a copy of the report named **List Product Profitability**.

► **Task 5: Expand the height of the report**

- Resize the report body to have a height of **5 inches (12.5 cm)**.
- Move the table data region to the bottom of the report.

► **Task 6: Add a Category List**

- Add a list data region to the top of the report.
 - Resize to **1.5 inches (4 cm)** high.
 - Position the top of the list to be **0.25 inches (0.5 cm)** below the top edge of the report.
- Add grouping and sorting to the list data region.
 - Open the **Tablix Properties** for the list data region and set the **Dataset name** value to **DataDetail**.
 - Open the **Group Properties** and set the name of the group to **Category**.
 - Set the **Group on** expression to **Category**.
 - Set the **Sort on** expression to **Category**.

► **Task 7: Add fields to the Category list**

- Using the **DataDetail** dataset, add the following fields horizontally starting in the bottom left corner of the **Category list** data region.
 - Category, 1.875 inches (5 cm) wide.
 - SalesAmount, 1.3 inches (4 cm) wide.
 - OrderQuantity, 1.3 inches (4 cm) wide.
- Set the format of the **Sum(SalesAmount)** text box to **C0**.
- Set the format of the **Sum(OrderQuantity)** text box to **N0**.

► **Task 8: Add header and footer rows**

- Copy the three text boxes from the list data region and paste them both above, and below the list to form the basis of header and footer rows.
- Format all six text boxes as follows.
 - Font size 12.
 - Black background.
 - White foreground.
- Alter the text in the six text boxes as follows.
 - In the **First(Category)** header text box, replace the text with **Category**.
 - In the **First(Category)** footer text box, replace the text with **Grand Total**.
 - In the **Sum(SalesAmount)** header text box, replace the text with **Sales Amount**.
 - In the **Sum(OrderQuantity)** header text box, replace the text with **Order Quantity**.
- Preview the report. Notice that the list contains all product categories together with a grand total.



Note: If you receive warnings about overlap then move the appropriate textboxes to remove the overlap and preview the report again.

► **Task 9: Add SubCategory list**

- Add a list data region beneath the category list data region.
 - Position the top of the list to be 0.25 inches (0.5 cm) below the category list data region.
 - Resize the list to the same width as the category list, and 0.25 inches (0.5 cm) high.
- Add grouping and sorting to the list data region.
 - Open the **Tablix Properties** for the list data region and set the **Dataset name** value to **DataDetail**.
 - Open the **Group Properties** and set the name of the group to **SubCategory**.
 - Set the group to group on **SubCategory**.
 - Set the group to sort on **SubCategory**.

► **Task 10: Add fields to the SubCategory list**

- Using the **Report Data** pane, add the **SubCategory** field to the subcategory list data region.
 - Position the text box **0.25 inches (0.5 cm)** from the left edge inside the subcategory list.
 - Resize to **1.75 inches (4.5 cm)** wide.
- Copy the **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes from the **Category** list and place them in the **SubCategory** list, just to the right of the **First(SubCategory)** text box.
 - Adjust the horizontal alignment as required.
- Preview the report. Notice that there are two lists: the first has a header and footer and contains a list of categories; the second contains a list of subcategories.

► **Task 11: Combine the Category and SubCategory lists**

- Move the subcategory list inside the category list.
 - Position the subcategory list just below the row containing the **First(Category)**, **Sum(SalesAmount)** and **Sum(OrderQuantity)** calculated cells.
- Copy the text boxes in the first row of the category list to a new row of text boxes under the subcategory list (the category list should contain two duplicate rows of text boxes, one above the subcategory list, and one below it).
- Remove the **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes in the first row of the category list (these values are now displayed below the subcategory list).
- Replace the expression in the first text box in the last row of the category list (just above the Grand Total cell), with the text **Category Total**.

► **Task 12: Delete the table version of the report**

- Delete the table data region.
- Save and then preview the report.
- Adjust the formatting of the report as necessary.
- Turn off 6236A-NY-SQL-01 virtual machine and discard changes.

Results: After this exercise, you should have successfully added category lists, header and footer rows, subcategory lists, and combined the category and subcategory lists.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips

Review Questions

1. When should you use a matrix?
2. What chart types might you use in your organization?

Common Issues Related to Interactive Navigation

Issue	Troubleshooting tip
Images and charts do not load when using Run As to run a report	
An unexpected error occurred during report processing on the server	
Large reports take too long to render	
Extra or missing white space when viewing or exporting a report	

Module 4

Manipulating Data Sets

Contents:

Lesson 1: Defining Report Data	4-3
Lesson 2: Using Parameters and Filters	4-7
Lesson 3: Using Parameter Lists	4-12
Lab: Manipulating Data Sets	4-21

Module Overview

- Defining Report Data
- Using Parameters and Filters
- Using Parameters Lists

In this module, you will explore using data sets to allow parameters in the reports you create. These parameters permit the user to provide selection criteria for a report, either by typing it into a text box, or by selecting possible values from a list. The values are then used to filter the data presented in the report so that the user sees only relevant data instead of all of the data that the report may contain.

You will also learn how to dynamically modify the data set so that parameter list values are always up to date for the user. This will eliminate the need to modify the report parameter lists every time the data in the database is changed.

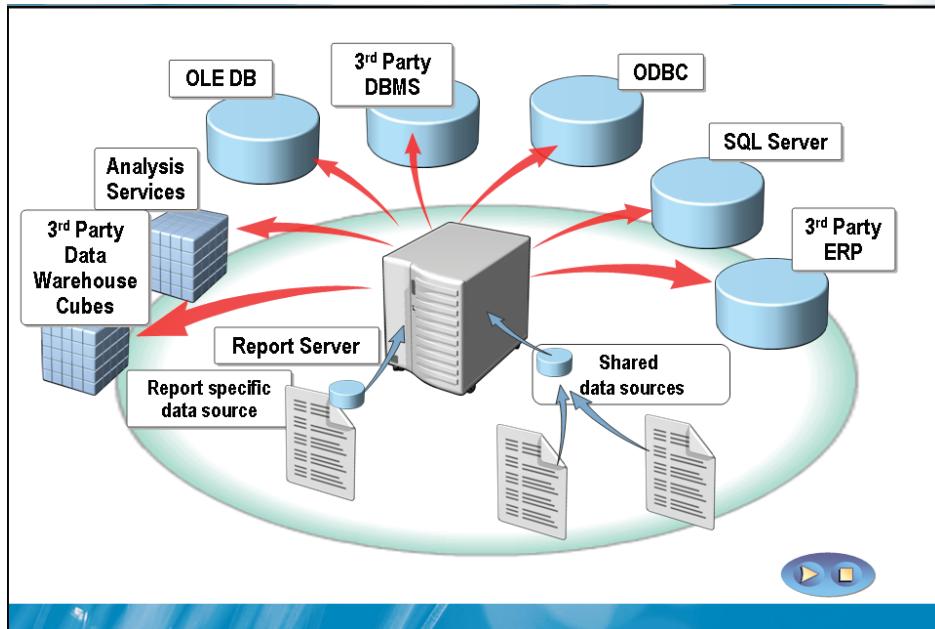
Lesson 1

Defining Report Data

- Connecting To a Data Source
- Querying a Data Source

Reporting Services in Microsoft® SQL Server® 2008 can query and display data from any of the multiple databases that exist in an organization today. In this lesson, you will learn about the methods available to attach to those databases and to set up queries to extract the data.

Connecting to a Data Source



Key Points

Reporting Services provides the ability to use the most common databases for data sources. These include: SQL Server, ODBC, OLE DB, Analysis Services and 3rd party database, data warehouse, and ERP systems.

- Reporting Services uses standard data providers as well as the faster SQL Server 2008 data providers.
- Relational and Multi-Dimensional databases can be used in Reporting Services.

Querying a Data Source

SQL Statements	Data sources use standard SQL queries to determine the structure and content of a data set
Query parameters	Parameterized queries provide flexibility based on user input or other values
Embedded Data Sources	Stored in the definition of the report and cannot be shared
Shared Data Sources	Stored on the report server and can easily be used by multiple reports

Key Points

A Transact SQL query is used to retrieve rows of data from the data source. These queries define the structure of the data that will be returned in a data set. Data source queries can also contain criteria to narrow the data set down to specific data required for a report.

Some things to think about when querying data sources:

- Report parameters that contain user input or values can be passed to the query.
- Queries are processed by the data source on the server.
- Data sets that are returned can be further refined to tailor the data for specific reports.

Demonstration: Creating a Data Source

In this demonstration, you will see how to:

- Create a data source
- Query a data source to create a data set

Question: When creating a data source you can select whether it is embedded or use a shared data source. What advantages and disadvantages do you see for each? Is there one that you anticipate using more than the other? Why?

Lesson 2

Using Parameters and Filters

- Report Parameters
- Using Report Parameters
- Using Query Parameters
- Filters

Parameters are a report feature that allows the user to input or select values that are used by a data source or a filter to refine the data available for display on a report. Parameter values are either passed to a query on the data source or used locally by the report in a filter.

Report Parameters

- **Pass values into a query parameter**
- **Filter data in the data set**
- **Pass values to a linked report**
- **Are used to create cascading parameter lists**
- **Are used to alter the appearance of a report**

Key Points

Report parameters give the developer flexibility to take an initial data set and further refine it to suit specific reporting needs.

Some features of report parameters include:

- They are used in the data set not the data source.
- Developers can set report parameters visibility to allow or disallow user input of values.
- Default values can be set to report parameters.
- When using a URL to access a report, parameter values can be passed directly to the report in the URL address.

Using Report Parameters

- Steps for creating report parameters**
- 1 Define name and data type**
 - 2 Use prompts**
 - 3 Specify parameter options such as hidden or allow null**
 - 4 Specify available values**
 - 5 Set a default value**

Key Points

The use of report parameters simplifies report development and deployment. By using report parameters one report can be used by multiple users while giving each user the option to select the specific data that he or she requires.

Report parameters have many options that can be selected to customize the behavior of the parameter or the way values are selected.

- Data type determines presentation method of parameter values.
- Parameters can be set to accept null values.
- A custom list of static values can be created for each parameter.
- Default values can be setup for each parameter.

Using Query Parameters

- **Query parameters and report parameters can be linked**

SQL query parameter

```
SELECT * FROM Sales  
WHERE Office = @Office  
AND Salesperson =  
@UserID
```

Parameters collection

```
Parameters!Office.Value  
User!UserID
```

- **Renamed query parameters do not automatically update the report parameter**

Key Points

Query parameters are used by the data source to allow user input of values for use in the selection of data for a report data set.

Query parameters:

- Are processed on the data source.
- Allow parameter values to be passed in from a report parameter.

Filters

An alternative approach to using parameters

- Restrict visible data after it has been retrieved
- Can be applied to data set, data region, or grouping
- Compare an expression to a value in the Filters tab of the data region

Expression	Operator	Value
=Fields!Category.Value	=	=Parameters!Category.Value

Key Points

Report data sets are further refined through the use of filters. Filters are used to examine the data based on criteria that can be related to data values or aggregates, and are used to refine data or grouping in a report.

Filters are used to refine the data presented by a report.

Filters:

- Are used to restrict visible data after it has been retrieved.
- Can filter data within a data set, a specific data region, and within groups.
- Can use expressions to filter data within a data region against set values.

Lesson 3

Using Parameter Lists

- Data-bound Parameters
- Adding Parameters with a Stored Procedure
- Dynamic Parameter Lists
- Adding “All Items” to a Parameter List
- Creating a Multi-Select Parameter List

Lists of parameters are often used to present possible values in place of straight user input, providing greater ease of use. Parameter lists can be populated manually with pre-set values at design time or at runtime using actual values from a query.

Parameter lists provide a way to present customized sets of values for use in queries and filters. The listed values are typically based on a query that uses a value from another parameter in the report.

Data-bound Parameters

- Displays a list of available values based on a query
- A special data set is typically created for this purpose
- Allows the available parameter values to dynamically change as the database changes

Key Points

Data-bound parameter values are populated by a query using data contained in the data set. Some examples include: only the salespeople within a give region, only the items within a specific category, postal codes within a region, and the like.

- Parameter values are retrieved directly from the data set.
- A query is used to populate a data-bound parameter. A value from another parameter is often passed to filter the query.

Question: Some reports are more suited for data-bound parameters than others. Which reports in your organization already use data-bound parameters? Which reports do you think will benefit by creating data-bound parameters?

Adding Parameters with a Stored Procedure

Stored Procedures:

- Abstract the data schema
- Allow extended logic
- Enhance security and reusability
- Must return a single data set

Key Points

Stored procedures are a common method for populating a parameter list with values. The data set returned by a stored procedure can contain data derived through complex calculations or logic flows that are not possible using a standard T-SQL query.

Using stored procedures to populate a parameter list allows for sophisticated calculations and logic flows to select or create values. Security can also be taken into consideration with stored procedures so that user or group permissions can be used as part of the criteria.

Advantages of stored procedures include:

- Stored procedures are stored in the database allowing multiple reports to share them.
- Stored procedures are pre-compiled to speed the processing time for complex procedures.

Question: Stored procedures offer flexibility to create lists based on complex calculations and logic. Does your organization use reports that would benefit or be enhanced significantly if a stored procedure is used to create a parameter list?

Dynamic Parameter Lists

First parameter value affects another parameter's list

- Add first parameter to query for second parameter's data set
- Put parameters in sequential order
- Example:
 - Enter country first, then select region from list

```
SELECT Region FROM Geography  
WHERE Country = Parameters!Country.Value
```

Key Points

Dynamic parameters provide a mechanism for changing the criteria of a query to load possible values that are more appropriate to a user's needs based on the value or values selected in another parameter. For instance, you may design a report to use the selection in a state parameter to drive the selection of possible values in a salesperson parameter list.

- Dynamic parameter list values represent actual values within the data set.
- Values in a dynamic parameter list are designed to change based on the value or values of another report parameter.
- It is important that the definitions of the parameters are listed in sequential order with the driving parameter definition listed first.

Adding "All Items" to a Parameter List

Sometimes a filter should include all possible items

- For parameter list query, create UNION query to add single item with constant

```
SELECT '[All Items]' AS City  
UNION  
SELECT DISTINCT City FROM Person.Address
```

- For main data set query, add OR condition for "All Items" parameter

```
SELECT *  
FROM Person.Address  
WHERE (City = @city) OR (@city = '[All Items]')
```

Key Points

"All Items" is a special case that alters the criteria in a parameter list query to populate the parameter list with all possible values in the data set. If the "All Items" option is not used, the parameter list will only contain a subset of the data contained within the data set.

- There are times when all possible values need to be displayed in a dynamic parameter list. By making a simple change to the query that selects the data for the list the query can be used to retrieve all possible values. In the previous example a territory is used to display only the salespeople for that territory. The only way to display all of the salespeople is to use the "All Items" option in the query.
- If the "All Items" option is not used the dynamic parameter lists present a filtered set of values.
- By using the "All Items" option the query can be told to present all possible values when needed.

Question: The "All Items" option is used to present all possible options in a parameter list. Although many reports benefit from this option there are some that will have too many parameter values to be practical. Which reports in your organization do you think will benefit from adding the "All Items" option? Which reports would not?

Create a Multi-Select Parameter List

Parameter lists can support multiple selections

- Allow user to select multiple values from a combo box
- Use an IN clause with a parameter

```
SELECT *
FROM Person.Address
WHERE City IN (@city)
```

Key Points

Reporting Services offers a parameter list option that allows you to select multiple values for use in a query. All of the selected values are then passed to the query for processing.

Special considerations include:

- Selecting multiple values creates a list of those values within a parameter.
- The entire list is passed to the query or filter.
- The WHERE clause in the Transact SQL statement must use the IN operator to evaluate the list.

Demonstration: Manipulating Data with Parameters

In this demonstration, you will see:

- How report parameters are used to filter data
- How parameters are setup to dynamically control the values of another parameter
- How to change a dynamic parameter to present all possible values

Question: When designing reports, what do you need to take into account when deciding which kind of parameter to create each of the following parameters?

Filter parameter (values are entered directly)? Parameter List? Dynamic Parameters? 'All Items' parameter list option?

Lab: Manipulating Data Sets

- Exercise 1: Using Parameters to Restrict Query Results
- Exercise 2: Using Parameters to Filter Report Data
- Exercise 3: Creating Dynamic Parameter Lists
- Exercise 4: Using Parameters with a Stored Procedure
- Exercise 5: If Time Permits: Displaying All Items in a Parameter List

Logon information

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

Estimated time: 60 minutes

Exercise 1: Using Parameters to Restrict Query Results

Scenario

The AdventureWorks team has asked you to enhance a number of existing reports.

The Product Profitability report uses a table to display total sales performance by product and summarized totals by product subcategory and category. To restrict data in the report by month and year, you need to create hard-coded parameters to filter the report.

In this exercise you will create parameters for use in a query. The parameters will be set with a list of possible choices and used in the query to select sales data based on month.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the data manipulation solution.
3. Modify the data set query.
4. Edit the **Year** and **Month** parameters.
5. Save and preview the report.

► **Task 1: Set up the lab environment**

- Start NY-SQL-01 and log on as **Administrator** using the password **Pa\$\$w0rd**.
- Run E:\MOD04\Labfiles\Starter\Exercise01\runScripts.bat.

► **Task 2: Open the data manipulation solution**

- Open **SQL Server Business Intelligence Development Studio**.
- Open the E:\MOD04\Labfiles\Starter\Exercise01\Data Manipulation.sln project file.
- Display the **Product Profitability.rdl** report, and then click the **Preview** tab.

► **Task 3: Modify the data set query**

- Edit the **Data Detail** query to change the static Year and Month values to parameters **@Year** and **@Month**.

► **Task 4: Edit the Year and Month parameters**

- Expand **Parameters** and edit the **@Year** parameter to have a default value of **2003**.
- Edit the **@Month** parameter to include the following specified values:

Label	Value
Jan	1
Feb	2
Mar	3
Apr	4
May	5
Jun	6

► **Task 5: Save and preview the report**

- Save the project.
- Change to the **Report Definition** tab, and then click **Preview**.
- Select **Jun** in the **Month** parameter list, and then click **View Report**.

Results: After this exercise, you should have created parameters and set values for selection.

Exercise 2: Using Parameters to Filter Report Data

Scenario

Your supervisor has asked you to change the report to use a parameter to filter the report data by category.

In this exercise, you will use the Category parameter to filter the data set for a tablix data region on the report.

The main tasks for this exercise are as follows:

1. Open the data manipulation solution.
2. Add a data set for the Category parameter.
3. Create the Category parameter.
4. Add a filter to the table data region.
5. Remove the Category group page break.
6. Add the Category to the report header.

► Task 1: Open the data manipulation solution

- Open SQL Server Business Intelligence Development Studio.
- Open the E:\MOD04\Labfiles\Starter\Exercise02\Data Manipulation.sln project file.
- Display the **Product Profitability.rdl** report, and then click the **Preview** tab.

► Task 2: Add a data set for the Category parameter

- Add a new data set to the **AdventureWorksDW** data source.
- Use the following code to create the query for the new **Category** data set:

```
SELECT DISTINCT EnglishProductName AS Category  
FROM DimProductCategory
```

- Run the query in Query Designer and check for errors.
- Add **Category** as the name and description values for the data set.

► **Task 3: Create the Category parameter**

- Add a parameter called **Category** to the parameters list.
- Use the following settings:

Label	Value
Data type	Text
Allow blank values	Unchecked
Available values	Get value from a query
Dataset	Category
Value	Category
Label	Category

- Add a default value of **Components**.
- Close the **Report Parameters** dialog box.

► **Task 4: Add a filter to the table data region**

- On the **Design** tab right-click the tablix, and then click **Tablix Properties**.
- In **Filters**, add the expression:

```
=Parameters!Category.Value
```

- Close the **Tablix Properties** dialog box.

► **Task 5: Remove the Category group page break**

- Right-click the **Category Footer** row and then click **Properties**.
- On **Page Breaks**, clear **Between each instance of a group** and click **OK**.
- Save the project.
- Preview the report using **Bikes** as the **Category** parameter value.

► **Task 6: Add the Category to the report header**

- On the **Design** tab add a text box about the graphic and resize it.
- Type the following expression into the field Expression property:

```
=Parameters!Category.Value
```
- Save the project.
- Preview the report with **Mar** selected for the Month and **Bikes** selected for the Category.
- Close the project.

Results: After this exercise, you should have created parameters and used them to filter report data.

Exercise 3: Creating Dynamic Parameter Lists

Scenario

The Product Details report uses a table to display product detail information. The report is filtered by a product subcategory report parameter. You will create a dynamic parameter list to filter the report based on both the product category and subcategory.

In this exercise, use the selected value in the Category parameter to alter the query used by the SubCategory parameter so that only subcategories of the selected category are listed as values.

The main tasks for this exercise are as follows:

1. Open the dynamic parameters solution.
2. Add a dataset for the Category parameter.
3. Modify the SubCategory data set query.
4. Modify the ProductDetail data set query.
5. Edit the ProductCategory parameter.
6. Set the title of the report.

► **Task 1: Open the dynamic parameters solution**

- Open SQL Server Business Intelligence Development Studio.
- Open the E:\MOD04\Labfiles\Starter\Exercise03\Dynamic Parameters.sln project file.
- Display the **Product Details.rdl** report, and then click the **Preview** tab.

► **Task 2: Add a dataset for the Category parameter**

- Add a new data set named **Category**.
- Use the **DataSet1** data source.
- Use the following code for the data set query:

```
SELECT ProductCategoryID, Name  
From Production.ProductCategory
```

- Run and test the query in **Query Designer**.
- Close the **Dataset Properties** dialog box.

► **Task 3: Modify the SubCategory data set query**

- In the **Dataset** list, open the **Properties** dialog box for **SubCategory**.
- Change the SQL statement to include a WHERE clause. The new statement will look like this:

```
SELECT ProductSubcategoryID, ProductCategoryID, Name  
FROM Production.ProductSubcategory  
WHERE (ProductCategoryID = @ProductCategory)
```

- Close the **Dataset Properties** dialog box.

► **Task 4: Modify the ProductDetail data set query.**

- Open the **ProductDetail Properties** dialog box.
- Add **PS.ProductCategoryID** to the columns list.
- Set the **ProductCategoryID** filter to:

```
= @ProductCategory
```

- Close the **Dataset Property** dialog box.
- Save the project.

► **Task 5: Edit the ProductCategory parameter**

- Move the **ProductCategory** parameter above **SubCategory** in the **Parameters** list.
- Open the **ProductCategory** parameter **Properties** dialog box.
- Verify that the data type is **Integer**.
- Change the settings as below:

Label	Value
Available Values	From query
Dataset	Category
Value	ProductCategoryID
Label	Name
Default Values	Specify values
Default values field	1

- Click **OK**.
- Close the **Report Parameters** dialog box.

► **Task 6: Set the title of the report**

- On the **Design** tab edit the expression for the **[&ReportName]** text box to read:

```
=Globals!ReportName + " for " & Parameters!ProductCategory.Label
```

- Save the project.
- On the **Preview** tab, select **Clothing** for the **ProductCategory**, and **Shorts** for the **SubCategory** parameters.
- View the report.
- Close the project.

Results: After this exercise, you should be able to use dynamic parameters to display a different list of available values in the second parameter based on the selected value in the first parameter.

Exercise 4: Using Parameters with a Stored Procedure

Scenario

You have been asked to use a stored procedure name sp_ActualVsQuota to supply the data for a report. You need to set up two input parameters to control the data set; Calendar Year and Group.

In this exercise, you will create a parameter that uses the sp_ActualVsQuota stored procedure to populate Detail parameter.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the stored procedure solution.
3. Create a report data source.
4. Create a stored procedure data set.
5. Edit report parameters.
6. Save and preview the report.

► **Task 1: Set up the lab environment**

- Run E:\MOD04\Labfiles\Starter\Exercise04\runScripts.bat.

► **Task 2: Open the stored procedure solution**

- Open SQL Server Business Intelligence Development Studio.
- Open the E:\MOD04\Labfiles\Starter\Exercise04\Stored Procedure.sln project file.
- Display the [Actual Vs Quota.rdl](#) report, and then click the Preview tab.

► **Task 3: Create a report data source**

- Add a new data source named **AdventureWorks2008**.
- Select **Use shared data source reference** and choose **AdventureWorksDW2008**.
- Close the **Datasource Properties** dialog box.

► **Task 4: Create a stored procedure data set**

- Add a new dataset named **Detail**.
- Select **AdventureWorks** for the **Data source**.
- For **Query type** select **StoredProcedure**.
- In the Query Designer select the **sp_ActualVsQuota** stored procedure.
- When the **Define Query Parameters** dialog box opens, type **2003** into the **@CalendarYear Parameter Value** text box, and **Europe** in the **@Group Parameter Value** text box.
- Close the **Query Designer** and the **Dataset Properties** dialog boxes.

► **Task 5: Edit report parameters**

- Edit the **CalendarYear** parameter to have no **Available values** and a default value of **2003**.
- Edit the **Group** parameter to have no **Available values**, and a default value of **North America**.

► **Task 6: Save and preview the report**

- Save the project.
- On the **Design** tab, click the **Tablix**, and then assign **Detail** as the **Dataset name**.
- Into the **Group** parameter box, type **Europe**.
- View the report.
- Close the project.
- If enough time remains, continue to exercise 5.
- If lab is complete, shutdown **NY-SQL-01** and discard changes.

Results: After this exercise, you should have created data sets and assigned default values to report parameters.

Exercise 5: If Time Permits: Displaying All Items in a Parameter List

Scenario

You have been asked to create a report that allows the user the option of selecting all of the categories from the parameter list.

In this exercise, you will change a parameter list to allow the display of all possible subcategories. When the report is finished the user will be able to select a specific category or the "All Items" option in the category list.

The main tasks for this exercise are as follows:

1. Open the all items solution.
2. Add all categories to the Category data set.
3. Update the DataDetail data set query.

► Task 1: Open the all items solution

- Open SQL Server Business Intelligence Development Studio.
- Open the E:\MOD04\Labfiles\Starter\Exercise05\All Items.sln project file.
- Display the All Items Product Profitability.rdl report and click the Preview tab.

► Task 2: Add all categories to the Category data set

- Edit the **Category** dataset to include the following code to create a UNION:

```
UNION SELECT -1, 'All Categories'
```

- The new statement will look like this:

```
SELECT DISTINCT ProductCategoryKey AS CategoryKey,
EnglishProductName AS Category
FROM DimProductCategory
UNION
SELECT -1 AS Expr1, 'All Categories' AS Expr2
ORDER BY CategoryKey
```

- Run and check the query in the Query Designer.
- Save the project.

► **Task 3: Update the DataDetail data set query**

- Edit the **DataDetail** parameter to include the following statement in the WHERE clause:

```
OR @Category = - 1
```

- The new statement will look like this:

```
SELECT Product, SubCategory, CategoryKey, Category, CostAmount,
SalesAmount, OrderQuantity, Month, MonthNumberOfYear, Year
FROM vProductProfitability
WHERE (Year = 2003) AND (MonthNumberOfYear IN (1, 2, 3))
AND (CategoryKey = @Category OR @Category = - 1)
```

- Save the project.
- On the **Preview** tab, in the **Category** parameter list, select **All Categories**.
- View the report and verify that all categories are displayed.
- Close the project.
- Shutdown NY-SQL-01 and discard changes.

Results: After this exercise, you should have successfully added the "**All Items**" option to the dynamic parameter to allow the display of all possible values for selection.

Module Review and Takeaways

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

Review Questions

1. In Lesson 1 we discussed accessing data sources. Reports can use either a report specific data source or a shared data source. In your organization, which do you anticipate using most often and why? Under what circumstances would you use the other type of data source?
2. A special parameter list option called "All Items" was introduced. When creating reports for your organization, what types of report data and reports do you anticipate using this option in? Which kinds of report data and reports do you think it would not be a good idea to include the "All Items" option? Why?

Common Issues

Issue	Troubleshooting tip
The graphical Query Designer screen is not available for creating a query in a report.	

Best Practices

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

- The use of shared data sources provides a consistent data source for all reports, offering a standard interface for querying and filtering the data.
- Provide at least one default value for a parameter when possible. This will allow a user to run the report immediately with the most common parameter values.

Real-world Issues and Scenarios

You can either discuss possible solutions for the scenarios at the end of the module or assign these scenarios as homework for the students. If you assign the scenarios as homework, you should provide some high-level hints that might help the students solve the problem stated in the scenarios. You can also do a follow-up debriefing the following day on some of the scenarios that have been assigned as homework to students.

- Scenario 1: The company VP of sales wants you to create a sales report by region with sales totals listed by Sales Manager, then by salesperson, then by date. What needs to be done in regard to data sets and parameters to allow him/her to select all sales managers and salespeople or narrow the report down to specific managers and salespeople?
- Scenario 2: Your supervisor asks you to edit a report to emphasize sales figures of sales managers that are below the company quota set by management. You decide to use a report filter based on a Quota value to change the formatting of the sales manager groups that meet the requirements. Where would you set the filter expression to accomplish this?

Tools

Tool	Use for	Where to find it
Graphical Query Designer	To build queries for parameters or filters	Click the Query Designer button in the Parameter dialog. If the graphical interface is not available click the Edit As Text button to change modes.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 5

Using Report Models

Contents:

Lesson 1: Creating Report Models	5-3
Lesson 2: Using Report Builder	5-10
Lab: Working with Report Models	5-19

Module Overview

- Creating Report Models
- Using Report Builder

In this module you will learn how to create custom report models and post them to the report server.

You will then learn the process of creating a report from the model using Report Builder. Report Builder is a report creation tool built into Reporting Services that allows a non-technical user to create on-demand reports using a simple drag-and-drop operation.

Lesson 1

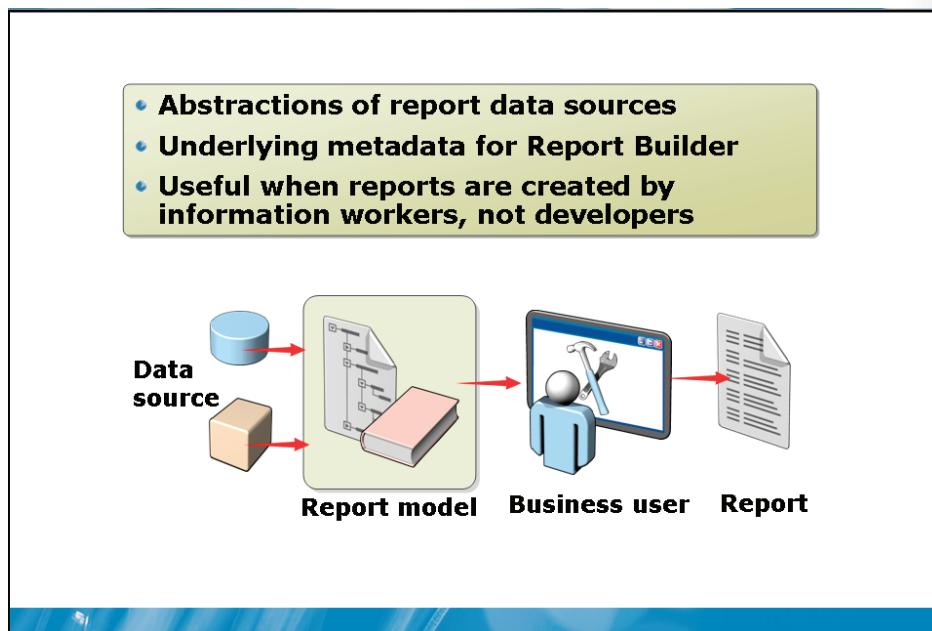
Creating Report Models

- What Are Report Models?
- Report Model Objects and Concepts
- Options for Creating Report Models
- Using Model Designer

Report models are pre-packaged sets of data, relationships, groupings, aggregates and more that can be deployed on the report server. Model Designer, Report Manager, and Microsoft® Office SharePoint® Server are used to develop the models.

You use the published models to simplify report creation by hiding the complex details such as database schema and entity relationships.

What Are Report Models?



Key Points

Report models are focused data sets created from a data source. They allow users to create reports without having to understand the underlying data and structure. You can simply select a model to work with and it will present you with the relevant data without worrying about relationships and other lower level details.

This is because:

- Models contain all metadata for the underlying data source.
- Relevant relationships are already specified.
- On-demand reports can easily be created using Report Builder.



Note: Report models are commonly used to create departmentally focused data sets. For instance, an organization may need to give a department the ability to quickly create on-demand reports without having to sort through data not relevant to their department.

Question: Within your organization, are there departments that frequently request custom or on-demand reports that would benefit from having a model available for them to query on a regular basis? What makes them good candidates for report models?

Report Model Objects and Concepts

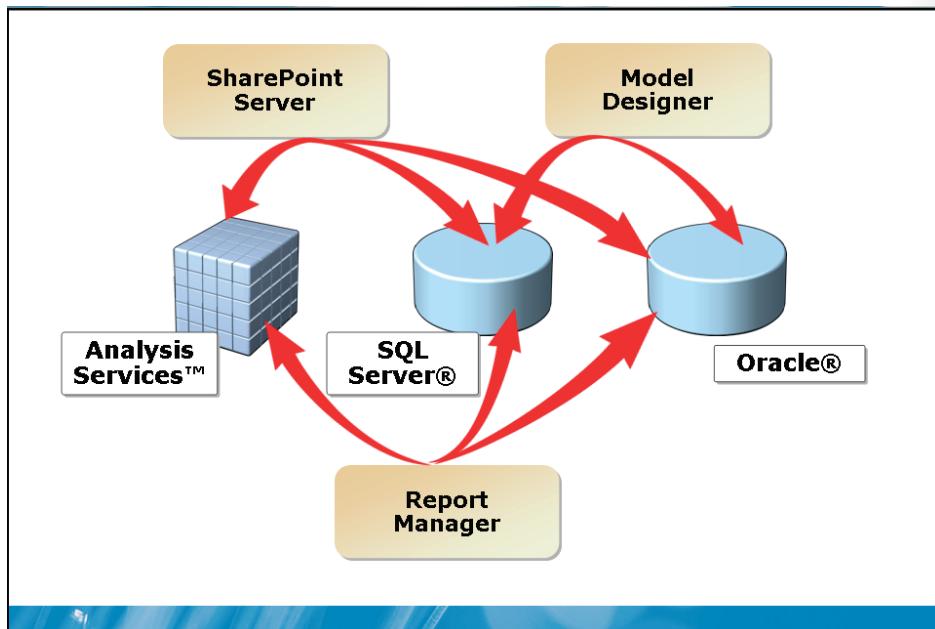


Key Points

Report models contain all of the information required to easily query data.

- Entities represent the tables and columns within the data set.
- Attributes are used to further refine the representation or behavior of an entity.
- Folders and perspectives allow grouping of entities and other folders and perspectives for special purposes or users.
- Roles define the relationships between entities.

Options for Creating Report Models



Key Points

There are three tools that can be used to create report models: Model Designer, Report Manager, and Office SharePoint Server. Each has its own strengths. Some of these tools may not be available within your organization.

- SharePoint Server offers a web based tool for creating models.
- Model Designer offers the most comprehensive tools for creating and controlling all aspects of a report model.
- Model Designer is the only tool that will not access Analysis Services cubes.
- Report Manager offers design tools directly accessible from the report server homepage.

Using Model Designer

Steps for creating a Report Model with Model Designer

- 1 Create a Report Model project**
- 2 Create a data source**
- 3 Create a data source view**
- 4 Define and refine the Report Model**
- 5 Publish the Report Model**

Key Points

Model Designer is part of the SQL Server Business Intelligence Development Studio (BIDS) and offers the most control and flexibility when creating a report model. After you create a model, you can use Model Designer to publish the model to the organization's report server for user access.

- Model Designer is most the comprehensive tool for creating report models.
- Use Model Designer to refine and maintain report models.
- Report models can be published to a report server using Model Designer.

Demonstration: Creating a Report Model

- Use Report Manager to create or select a data source
- Generate the model
- Publish the model

Question: What reasons can you think of to use an older or third-party data provider instead of the new ones shipping with SQL Server 2008?

Question: What should you consider when determining which tool to use for creating a report model?

Lesson 2

Using Report Builder

- What is Report Builder?
- Building a Report in Report Builder
- Working with Entities and Folders
- Grouping and Sorting Data
- Filtering Data
- Viewing and Publishing a Report

Report Builder is the drag and drop report creation tool available from the report server home page. You use Report Builder to create on-demand reports using the report models developed by Model Designer, Report Manager, or SharePoint Server.

In Report Builder, you can create reports and apply grouping, sorting and filtering to the report model data set. When the report is created it can be published to the report server for use by other users at a later time.

What Is Report Builder

Uses report models to facilitate the creation of ad hoc reports

A familiar Microsoft Office like environment for easily creating reports

Allows automatic generation of click through reports for exploring report data

Can be launched from SharePoint

Can be launched from third party applications for integrated reporting

Key Points

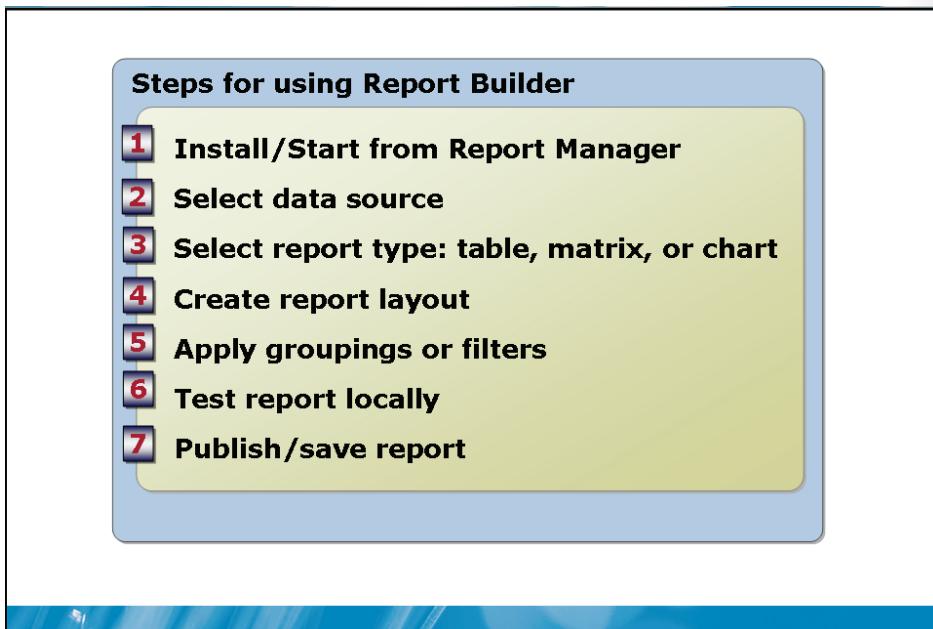
Report Builder is a tool that is launched from the report server and SharePoint as well as from third party applications that it has been integrated into. Report models are used by Report Builder to create reports and refine them for on-demand use or for publishing.

- Report models are the basis for all reports created using Report Builder.
- Report Builder can be integrated into third party applications.
- SharePoint Server and Reporting Services are the most common launch points for Report Builder.

An organization that uses SharePoint Server may wish to create and publish reports within the SharePoint environment. This offers a familiar web based environment for the average user to access Report Builder and the reports created using Report Builder.

Question: Knowing that Report Builder can be integrated with third party applications, what kinds of in-house applications exist in your organization that may benefit from having Report Builder?

Building a Report in Report Builder



Key Points

The basic steps to follow in building an on-demand report are listed above. By using Report Builder you are free to focus on formatting and manipulating the data to meet your unique reporting requirements.

- Start Report Builder by using Report Manager or a URL to a SharePoint site.
- You select the type of report you wish to create from a list of three common report types: table, matrix, or chart.
- Easily group, sort, and filter data in Report Builder.

Working with Entities and Folders

Explorer pane shows entities, fields, and folders

- **First entity dragged to report becomes primary report entity**
- **Available entities depend on the primary entity**
- **Use Advanced Mode to view additional relationship entities**

Key Points

Entities are the tables and fields in a report model's data set. The first entity dragged onto the report determines what other entities will be available to the report. Only entities related to the primary entity will be available.

- The primary entity provides the context for the report.
- Only entities related to the primary entity are shown in the Entities list.
- An advanced mode is available to allow non-typical entities to be listed, such as lookup entities.

Grouping and Sorting Data

Groups are added automatically based on the data relationships

Groups are based on values or entities

Sorting can be by fields within a group or for any field for which subtotals are displayed

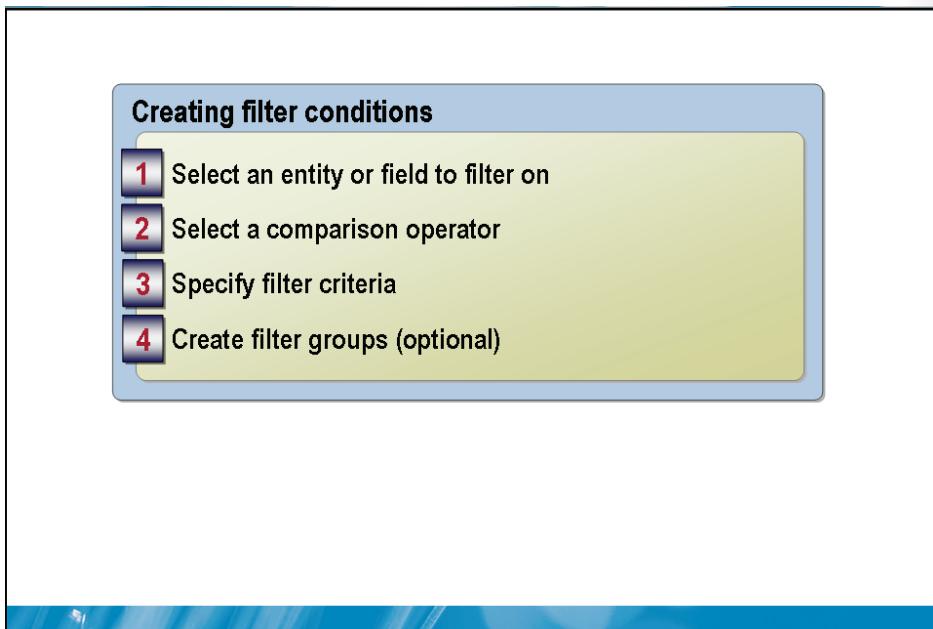
Grouping can be used to control page breaks

Key Points

Grouping and sorting organizes data on the report to allow for subtotaling and totaling of the data for more meaningful representation and analysis. Report Builder uses the entity relationships, or roles, to automatically sort, add groups, and subtotal the data in the report.

- Groups are automatically created based on the data relationships in the report.
- Groups can be based on entities or values.
- Grouping can be used to format reports with page breaks.

Filtering Data

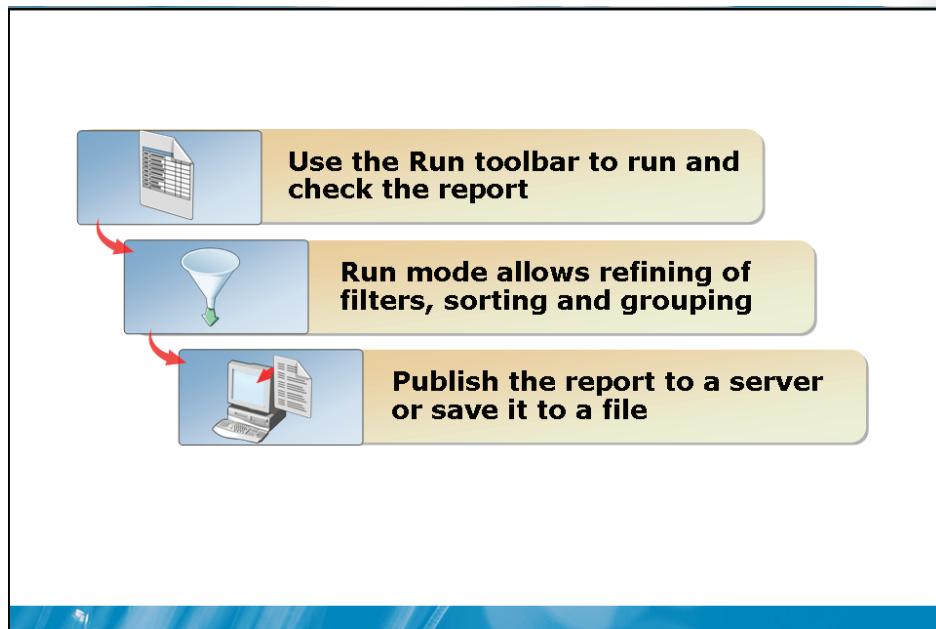


Key Points

Filters allow you to compare data values and use those comparisons to limit data or alter the formatting of the report.

- You can use filters to create subsets of the report data set.
- Filters can also be applied to groups of data.

Viewing and Publishing a Report



Key Points

Report Builder is used to run and refine your reports without having to publish them first. Once the report is ready to publish, you can use Report Builder to add the report to the report server for future use.

In Report Builder, you can:

- Use the **Run** toolbar to run reports and check them before publishing.
- In run mode you can refine the data filters, sorting and grouping.
- When you are finished creating the report and verifying it you can publish it to the report server from Report Builder.

Demonstration: Creating and Running a Report

- Use Report Builder to select a report model
- Select fields and entities from the model to display in the report
 - Define Sorting and Grouping
 - Create a filter for the report
 - Add formatting to the report
 - Run the report
 - Save the report to the Report Server
 - View the report

Question: In Report Builder, how did the selection of the contact entity fields affect the rest of the design of the report?

Lab: Working with Report Models

- Exercise 1: Creating a Report Model
- Exercise 2: Using Report Builder to Create and Execute a Report

Logon information

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

Estimated time: 45 minutes

Exercise 1: Creating a Report Model

Scenario

The sales managers at Adventure Works need to be able to create their own reports to show customer sales data. The sales managers are experienced Windows users, but they are not developers and don't want to use Business Intelligence Development Studio to create their reports. You have therefore decided to create a report model that they can use in Report Builder when they create their reports.

The management at Adventure works has asked you to create an easy way for customer sales reports to be created by departmental staff. They have requested that you look into report models as a possible solution.

In this exercise you will create a report model project containing customer and sales order data from the AdventureWorks Analysis Solution and publish the model to the report server.

The main tasks for this exercise are as follows:

1. Start the Virtual Machine.
2. Create a Report Model Project.
3. Create a Data Source.
4. Create a Data Source View.
5. Create a Report Model.
6. Publish the Report Model to a Report Server.

► **Task 1: Start the Virtual Machine**

- Start NY-SQL-01 and log on as **Administrator** using the password **Pa\$\$w0rd**.

► **Task 2: Create a Report Model Project**

- Open SQL Server Business Intelligence Development Studio and start a new Report Model project called **AdventureWorks Report Model** in E:\MOD05\Labfiles\Solution\Exercise01.

► **Task 3: Create a Data Source**

- Add a new data source on **NY-SQL-01**.
- Use the **AdventureWorks2008** database.
- Call the data source **Adventure Works2008**.

► **Task 4: Create a Data Source View**

- Create a new data source view based on the **Adventure Works2008** data source.
- Include **Person (Person)**, **Customer (Sales)**, and **SalesOrderHeader(Sales)**.
- Call the data source view **Customer Sales**.

► **Task 5: Create a Report Model**

- Generate a report model using the **Customer Sales.dsv** data source view.
- Rename the Sales Order Headers entity to **Sales Orders**.
- Rename the **#Sales Order Headers** attribute to **#Sales Orders**.
- Save the project.

► **Task 6: Publish a Report to a Report Server**

- Deploy the report model to <http://ny-sql-01/ReportServer>.
- Open the Report Manager and verify that the new **Customer Sales** model is on the model list.

Results: After this exercise, you should have created a report model, customized it for easier use, and published the model to the report server

Exercise 2: Using Report Builder to Create and Execute a Report

Scenario

Now that you have created the report model, management would like you to create the first report as an example. They would like the report to be available company wide.

In this exercise you will use the report Builder to create a report based on the Customer Sales model that you created. You will then publish the report to a report server.

The main tasks for this exercise are as follows:

1. Creating a Tabular Report.
2. Add Fields and Entities to the Report.
3. Define the Sort Order.
4. Define the Report Filter.
5. Format the Report.
6. Run the Report.
7. Save the Report to the Report Server.
8. View the Report.

► Task 1: Creating a Tabular Report

- Open Report Manager and start the **Report Builder**.
- Start a tabular report using the **Customer Sales** report model.

► Task 2: Add Fields and Entities to the Report

- Drag the **Last Name** field onto the design pane and change the formula to include the **First Name** to display the entire name in the field.
- Add **#Sales Orders** to the report.
- Add **Total Total Due** to the report and change the text value of the label to **Total Value**.
- Change the format to **\$1,345.67**.

- ▶ **Task 3: Define the Sort Order**
 - Under **Sort and Group** sort by **Total Value** descending.
 - Then, sort by **Name**.
- ▶ **Task 4: Define the Report Filter**
 - Add a filter for **Order Date** from 10/1/2003 to 12/31/2003.
- ▶ **Task 5: Format the Report**
 - Add the title **Customer Sales**.
 - Format the title with font color of **White** and fill color of **Cornflower**.
 - Move the filter description just below the title and stretch it to match the title width.
- ▶ **Task 6: Run the Report**
 - Run the report and verify that it runs as expected.
- ▶ **Task 7: Save the Report to the Report Server**
 - Save the report to the report server naming it **Customer Sales**.
 - Close the Report Builder.
- ▶ **Task 8: View the Report**
 - Open <http://ny-sql-01/reports>.
 - Re-sort the report by **Name**.
 - Display the report and show data for **Abercrombie, Kim**.
 - Close Microsoft Windows® Internet Explorer®.
 - Shutdown **NY-SQL-01** and discard changes.

Results: After this exercise, you should have created a report using a report model, formatted the report, published it to the report server, and viewed the report.

Module Review and Takeaways

- Review Questions
- Real-world Issues and Scenarios

Review Questions

1. In Lesson 1 you learned about creating and publishing report models. When defining a report model certain objects are created. What are some of these objects and how do they relate to the report model?
2. In Lesson 2 Report Builder was used to create and publish a report. What is the first consideration after a report model is selected when creating the report?
3. If you have an Analysis Services cube to create a report model for, which tool would not be suitable for creating the model?
4. When creating a report using a report model how does the report determine the primary entity?
5. Reports can be published to two different servers. One of them is Report Server. What is the other server that reports can be published to? What advantage does each have?

Real-world Issues and Scenarios

1. Adventure Works requires a sales report that summarizes all sales by territory. They also would like to be able to see the sales by sales person within a territory, and customer sales by sales person. The initial report must only show the territory totals within a date range and the other reports must be run with the same parameter values in the easiest way possible.
2. Adventure Works requires that each department have access to only the data that pertains to them.
3. Your organization has a number of users who frequently require special reports from the same data set.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 6

Publishing and Executing Reports

Contents:

Lesson 1: Publishing Reports	6-3
Lesson 2: Executing Reports	6-9
Lesson 3: Creating Cached Instances	6-14
Lesson 4: Creating Snapshots and Report History	6-20
Lab: Publishing and Executing Reports	6-30

Module Overview

- Publishing Reports
- Executing Reports
- Creating Cached Instances
- Creating Snapshots and Report History

Publishing reports makes them available to your organization's users. This can help when you have users in the organization who require access to reports for a particular application, but do not require access to the application itself.

You will learn about the various ways that reports are published and what type of report each of these are best suited for.

Reports can be published for execution on demand. They can also be published so that they are cached to free up valuable system resources. And they can be published to create point in time snapshots.

Snapshots can be added to report history for storage and retrieval at a later date.

This makes the reports accessible for use by any authorized user within the organization. Reports are accessed using Report Manager via Internet Explorer, a SharePoint site, and through subscriptions. The access method used will depend upon the particular needs of the user.

Lesson 1

Publishing Reports

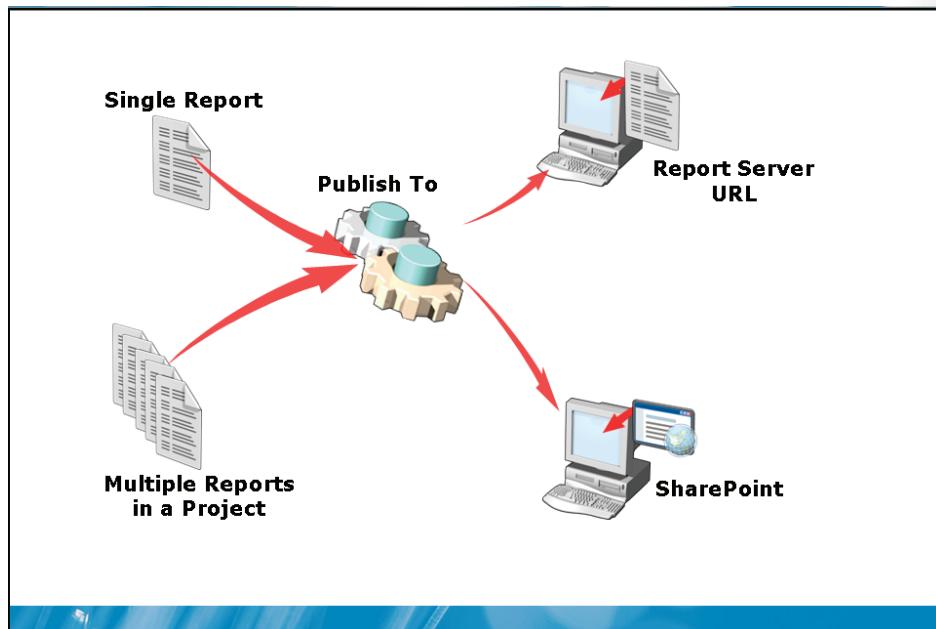
- Publishing with Report Designer
- Publishing with Report Manager
- Configuring Report Properties
- Updating Data Sources
- Updating Reports

When a report is published it is made available to all authorized users within an organization for viewing.

Reports can be published using Report Designer or Report Manager. In this lesson, you will learn how to publish reports and republish them when they are updated. Special attention will be given to the results of updating a report data source as opposed to updating the report definition itself.

You will also learn about the different report properties that can be set to fine tune report execution.

Publishing with Report Designer



Key Points

Report Designer is accessed through the Microsoft® SQL Server® Business Intelligence Development Studio. By using Report Designer, you can create single reports or projects that contain many reports, and then publish them singly or as a whole.

- Report Designer allows the creation of reporting projects with multiple reports grouped together.
- You can publish to a Report Server or Microsoft Office SharePoint® Server.
- Report Designer can publish a single report, group of reports, or all reports in the project.

Publishing with Report Manager

Upload report files

Rename or move a report

Modify a published report definition

Delete reports

Create linked reports

Key Points

Report Manager is a web-based interface that allows you to publish reports by uploading them directly to the Report Server.

Report Manager also allows you to organize the reports by placing them in folders, renaming them, deleting them, and creating linked reports.

Report Manager is:

- Designed to allow less technical users to create reports
- A web based reporting environment
- An environment that allows reports to be organized by placing them in folders

Configuring Report Properties

Property	Options
Data Sources	<ul style="list-style-type: none">• Can be shared or report specific• Specify database login credentials• Store credentials in report server database
Execution	<ul style="list-style-type: none">• Designate report execution frequency• Create execution schedules (shared and custom)• Determine report persistence
History	<ul style="list-style-type: none">• Determine how report history is retained
Parameters	<ul style="list-style-type: none">• Enable/disable parameter interactivity• Identify default value
Subscriptions	<ul style="list-style-type: none">• Select report delivery, for example email or file share
Security	<ul style="list-style-type: none">• Control how users interact with content

Key Points

Report properties control the way the Report Server and the users interact with the published reports. This is where data sources can be maintained and you can set properties that control how and when a report is executed.

- Reporting subscription delivery settings are maintained in the report properties.
- Report properties control how a report is executed, execution schedule and persistence.
- You can set default parameter values and enable and disable user interactivity for report parameters.

Question: A manager who uses a report on a daily basis approaches you. The report needs to be run at a specific time of day and she sometimes cannot run it at that time. What can you setup for the manager to ensure that the report runs at the required time and is available to her personally?

Updating Data Sources

Data sources can be updated to:

- **Change the data source name**
- **Change isolation level**
- **Point to a new server**
- **Change authentication method**
- **Point to a different database**

Key Points

Data sources can be updated without affecting the actual report definition. This allows flexibility and easy maintenance of the data source when the underlying database may have changed or the database is moved to another server.

Updating a data source does not change the report definition.

Data sources may require updating when:

- The server being pointed to is changed.
- The database is moved or a different database is going to be used.
- The authentication method is changed.

Updating Reports

Results of Modifying and Republishing Reports

- Replaces report definitions stored in report server database
- Does not overwrite parameter property changes
- Does not change execution, history, security, or subscription properties

Key Points

Updating of report definitions is a common maintenance task. Updates are required when:

- The underlying database is changed.
- The report users' requirements change.
- A layout change is requested.

Special care must be taken when updating published reports.

- Publishing an updated report definition will overwrite the existing report definition.
- Report Manager properties for the report will not be changed.
- Execution method, history, security, and subscriptions will not be changed.

Lesson 2

Executing Reports

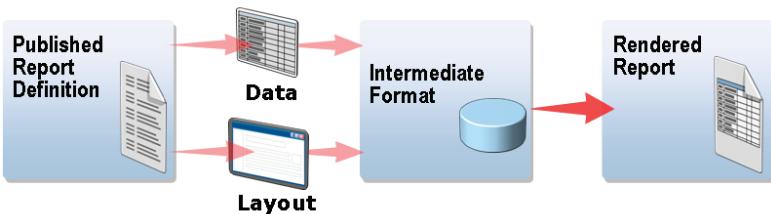
- Execution Process Flow
- Execute Reports on Demand
- Linked Reports

Once a report is published, it must be executed in order to view it. Published reports are executed when a user accesses the report for viewing through Report Manager or SharePoint. Upon execution, the data is queried and combined with the report layout for viewing.

The execution of published reports is controlled by setting properties for the report in the Report Manager. These properties determine whether a report can be executed on demand or on a schedule. You can also control whether execution of a report will create a cached instance as well as whether linked reports can be created for the report.

Execution Process Flow

- Execution is the process of turning a published report into a rendered report



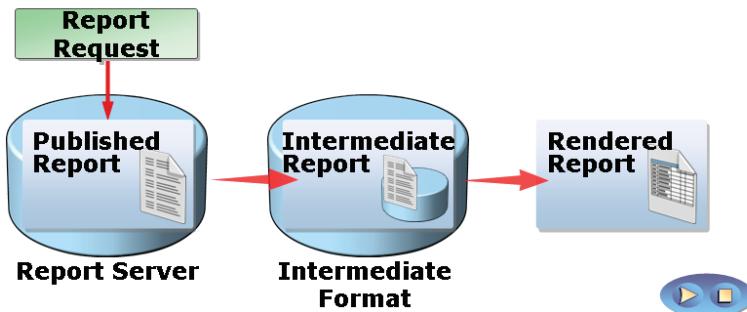
Key Points

When a report is executed, the Report Server uses the report definition to request a data set from the data source and to determine the layout of the data set for the report.

- The server combines the data set with layout information to create an intermediate format for a report.
- The intermediate format is what is stored for cached instances and snapshots.
- Rendering the report creates a viewable format that is specific to the browser or platform in which the report is being viewed.

Execute Reports on Demand

- Every report request triggers the same execution process:
 - Retrieves up-to-date data and processes report
 - Creates intermediate report and temporarily stores result in the session cache in ReportServerTempDB
 - Renders using intermediate report



Key Points

Reports executed on demand are executed by the user when needed. These reports contain the most current data set and are not cached.

- Execute on demand provides the most current data to the user.
- Reports used infrequently or require the most current data are the best candidates.
- Exploring data in a model based report is the same as executing a report on demand.

Linked Reports

Allow reuse of base report data source and report definition

- Alter linked report properties
 - Execution
 - Parameters
 - Subscriptions
 - Security
- Created in Report Manager

Key Points

Linked reports are reports that use the same data source as a report and present the data in a different manner. The difference could be alternate grouping, more detail, or summary data.

Linked reports appear in the parent report as links.

- Frequently used to display more detail.
- Linked reports use the same data source as the base report.
- Linked reports can have their own parameter properties.
- Linked reports can be saved as snapshots to report history.

Demonstration: Executing Published Reports

In this demonstration, you will see:

- How to publish reports to the Report Server
- How to browse the published reports in Report Manager
- How to update a published report

Question: When you update a published report, what are some of the considerations that you need to be aware of concerning updating the data source, as opposed to updating the report definition?

Lesson 3

Creating Cached Instances

- Executing Cached Instances
- Using Query Parameters with Cached Instances
- Using Filters with Cached Instances
- Configuring Cached Instances

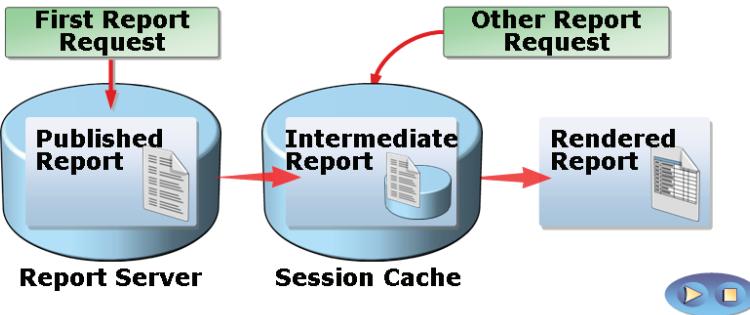
Report caching is a performance enhancing technique in Reporting Services that saves copies of reports for faster viewing.

Cached instances are intermediate forms of reports that are stored for a given time. These instances must have an expiration set to force a refresh of the data set used for the report.

Most suitable for caching are the reports that are accessed frequently and that place a higher burden on system resources.

Executing Cached Instances

- Execution process for cached instances
 1. Retrieves most up-to-date data and processes report
 2. Creates intermediate report and stores intermediate result in the cache in ReportServerTempDB
 3. Flags intermediate report as a cached instance
 4. Renders report from cached instance
 5. Later requests for the same report are retrieved from the cached copy



Key Points

Cached instances are reports that the server processes up to the intermediate format. These intermediate versions of the report are created at a specific time or when certain conditions are present, such as:

- Intermediate report formats are created and cached.
- The creation of a cached instance can be set to occur upon:
 - First use of the report.
 - On a scheduled basis.
 - Non-scheduled refreshes will occur if new parameter values are used.

Using Query Parameters with Cached Instances

Query parameters affect cached instance creation

- Parameters applied when the cached instance is created
- Changing parameter values results in new cached instance
 - Multiple cached instances can exist for the same report based on different parameters
 - Can result in many copies stored in the cache

Key Points

Query parameters are only processed by the server when the cached instance is created.

- Different report requests with different query parameter values will cause the server to cache an additional copy of the report for each unique set of parameter values.
- If different parameter values are used, then multiple copies of the report will be available for viewing.

Using Filters with Cached Instances

Filtering reports reduces cached instances

- Filters applied during each render
- Filters use current report parameter values
- Changing parameter values does not create new instance, just filters the cached instance

Key Points

Because filters are applied when a report is rendered they will not trigger the creation of a new-cached instance on the Report Server.

- Filters are applied to the current data set.
- Filters do not force a refresh of a cached instance.

Configuring Cached Instances

- **Cached instances are temporary and must expire**

- **Define interval**
- **Report-specific schedule**
- **Shared schedule**

- **No limit to the number of cached instances**
- **Expirations implemented by using SQL Server Agent**
- **Cached instances must have stored data source credentials**

Key Points

Because cached instances are temporary, you must remember:

- All cached instances must expire.
- Expiration can be set as a time interval or a schedule.
- Once a cached instance is expired the server will create a new instance based on the report setting.
- SQL Server Agent is used to schedule expiration of cached instances.
- Cached instances must use data sources with stored credentials.

Question: When using cached instances of a report, when might forcing expiration of the instance be a good idea?

Demonstration: Caching Report Instances

In this demonstration, you will see:

- How to configure a report for cached execution
- How to set a cached instance to expire based on a schedule

Question: When scheduling expirations for multiple reports what two options do you have for setting the schedule? When would you use each of them?

Lesson 4

Creating Snapshots and Report History

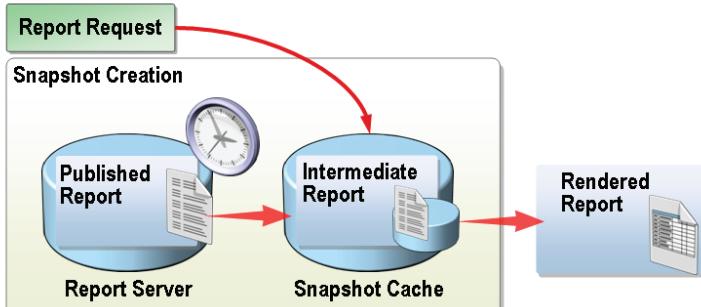
- Executing Snapshot Reports
- Using Query Parameters with Snapshots
- Using Filters with Snapshots
- Configuring Snapshots
- Using Report History
- Configuring Report History
- Cached Instances vs. Report Snapshots

Report Server allows you to create snapshots of reports and retain copies for future viewing. The snapshots can be scheduled like cached instances and you can set how many copies to retain, and how long to retain them.

These snapshots are handy when reports need to be automated, and possibly stored for future viewing. Automating reports may be required for various reasons. Some users may simply require a report to always be executed at a specific time. Snapshots allow you to schedule reports and have them available for viewing with the appropriate data. History keeps a sequence of snapshots available for later viewing.

Executing Snapshot Reports

- Execution process for snapshots
 - Creation of snapshot is scheduled for a specific point in time
 - The intermediate report is created and the result is stored as a snapshot in the report server database
 - Requests are satisfied by retrieving and rendering the snapshot



Key Points

Snapshots are intermediate forms of reports similar to cached instances. They are different in that snapshots are only run on a schedule. Snapshots are:

- Stored in intermediate form.
- Created at specific times.
- Retrieved and rendered when the report is requested for viewing.

Using Query Parameters with Snapshots

Only a default parameter can be used in a snapshot

Snapshots do not allow user input of different parameter values

Parameter values are only applied at the time the snapshot is created

Key Points

Since snapshots are created at a specific point in time it is not possible to collect user selected values for query parameters. Because of this query parameters must be set with default values for the data source query to use when creating the report data set.

- Query parameters must use default values.
- User input of parameter values is not possible in a snapshot.
- Parameter values are only used when the snapshot is created.

Question: Because snapshot reports cannot accept user input, what kinds of reports does your organization use that would be good candidates for creating snapshots?

Using Filters with Snapshots

Filters are only applied during browsing

Filters use the current snapshot parameter values

Filters only refine the existing data from the snapshot

Key Points

By using filters in a report instead of parameters, you can refine the data in a snapshot. Filters act upon the current snapshot data set when browsing a report.

- Filters are local to the report and do not require a new data source query.
- Filters use the current snapshot data set.
- Filters are applied only when browsing in a report.

Configuring Snapshots

Snapshots are executed manually or on a schedule

SQL Server Agent is used to schedule snapshots

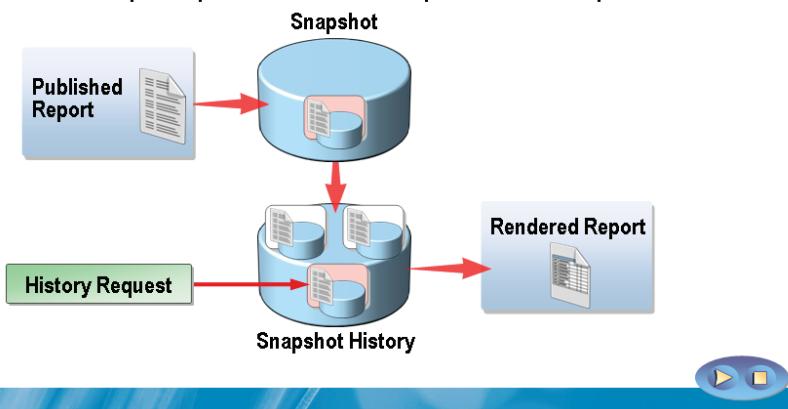
Key Points

When configuring snapshots keep in mind that:

- Execution schedules can be specific to the report or shared.
- Shared schedules are easier to track and maintain.
- Schedules can include a start and stop date.

Using Report History

- Snapshots are created when a published report is executed
- History stores a series of snapshots of the same report for future reference
- Specific history snapshots can be requested at any time



Key Points

Report history keeps multiple snapshots of a report for future viewing. These snapshots are searchable and any version of the report can be accessed when needed.

- Snapshots are stored in report history.
- Specific snapshots can be viewed at any time.
- Report history can be stored indefinitely.

Configuring Report History

- **Report history is not setup by default**
- **Report history options include:**
 - **Store all snapshots**
 - **Manually add snapshots**
 - **Schedule history snapshots**
 - **Limit number of snapshots**
- **Requires stored credentials**

Key Points

Report history must be configured manually for each report that you wish to keep history. Some things to remember include:

- You can use a report specific or shared schedule.
- Storage capacity is a concern for frequently created snapshots.
- You may want to limit the number of snapshots stored if the snapshots are executed frequently.

Cached Instances vs. Report Snapshots

•Cached Instances

- Enhances speed of processing
- Always has an expiration

•Report Snapshots

- Can be scheduled to run at off hours
- Assures that reports reflect data at a specific point in time
- Can be stored in report history
- Must use stored credentials

Key Points

Cached instances and report snapshots are similar in that they both keep a copy of a report for future viewing. However, cached instances are designed to speed up processing and always expire. Snapshots are designed to be kept available for a longer period and are executed to assure that the data set reflects the data as of a specific time.

- Cached Instances:
 - Enhances speed of processing
 - Always expires
 - Can be created on first use after expiration or on schedule
- Report Snapshots:
 - Can be created manually or on a schedule
 - Data is as of a point in time
 - Can be stored in report history

Question: Given the differences in how cached instances and snapshots operate, what reports can you think of in your organization that would work best in each of the two caching methods?

Demonstration: Using Report Snapshots and History

In this demonstration, you will see:

- How to prepare a report for snapshots by using a data source with stored credentials
- How to schedule a snapshot to run at the beginning of the month
- How to set snapshot retention rules
- How to view the snapshot history for a report

Question: Report history can retain an unlimited number of snapshots for future viewing. If your server has limited storage, what can you do to control the space used by stored snapshots?

Lab 6: Publishing and Executing Reports

- Exercise 1: Publishing Reports
- Exercise 2: Executing a Report on Demand
- Exercise 3: Configuring and Viewing a Cached Report
- Exercise 4: Configuring and Viewing a Snapshot Report

Logon information

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

Estimated time: 45 minutes

Exercise 1: Publishing Reports

Scenario

The Adventure Works team has asked you to publish a number of pre-written reports to a report server. You must configure report delivery so that the performance of the report server is maintained while ensuring that each report is delivered in a timely fashion.

You have been given a complete report solution by the Adventure Works developers. They have requested that you publish the entire solution to the report server and test the reports to be sure they are configured properly.

In this exercise, you will deploy the AdventureWorks solution containing multiple reports to the Report Server.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open the solution.
3. Deploy the solution.
4. Browse the published reports.
5. Update the Product Profitability report.
6. Browse the updated Product Profitability report.

► **Task 1: Set up the lab environment**

- Start the **6235A-NY-SQL-01** virtual machine and log on as **Administrator** with the password **Pa\$\$w0rd**.
- Run E:\MOD06\Labfiles\Starter\Exercise01\runScript.bat.

► **Task 2: Open the solution**

- Open **SQL Server Business Intelligence Development Studio**.
- Open the E:\MOD06\Labfiles\Starter\Exercise01\AdventureWorks \AdventureWorks.sln project.
- Examine the three projects and their respective reports.

► **Task 3: Deploy the solution**

- Change the **TargetServerURL** to **http://ny-sql-01/ReportServer**.
- Deploy the **AdventureWorks** solution.
- Confirm that there are no warnings in the output window.
- Keep Business Intelligence Development Studio open.

► **Task 4: Browse the published reports**

- Open Microsoft® Internet Explorer® and browse to **http://ny-sql-01/reports**.
- In the **Product Sales** folder browse to the **Product Profitability** report.

► **Task 5: Update the Product Profitability report**

- Open the **Product Profitability.rdl** in Business Intelligence Development Studio.
- Add a Textbox to the **Page Footer** next to the **Company Confidential** text box and resize it to the right edge of the page.
- Type **=Globals!ExecutionTime** in the text box.
- Format the text box as Italic and right aligned.
- Preview the report and notice the new text box contents.
- Deploy the **Product Profitability** report and note the warning message.
- In the **Product Sales** project properties change **OverwriteDataSources** to **True**.
- Deploy the **Product Profitability** report and note that no warnings are displayed.
- Close Business Intelligence Development Studio.

► **Task 6: Browse the updated Product Profitability report**

- In Internet Explorer browse the **Product Profitability** report.
- Keep Report Manager open.

Results: After this exercise, you should have published a report solution, made updates to a report in the solution and republished the report.

Exercise 2: Executing a Report on Demand

Scenario

Now that the solution has been published, the management would like you to make sure the Order Details report is set so that anyone who runs the report will receive the most current data.

In this exercise you will configure a published report to be executed on demand.

The main tasks for this exercise are as follows:

1. Configure the Order Details report for on-demand execution.
2. View the Order Details report.

► **Task 1: Configure the Order Details report for on demand execution**

- In Report Manager, in the **Reseller Sales** folder, go to the **Order Details** report.
- In the **Connect Using** property box under **Properties and Data Sources**, select the checkbox for **Use as Windows credentials when connecting to the data source**.
- Under the **Execution** link confirm that the report is set to always run with the most recent data.

► **Task 2: View the Order Details report**

- Go to the **View** tab and login as **NY-SQL-01/Student** with the password of **Pa\$\$w0rd**.
- View and browse the report.
- Keep Report Manager open.

Results: After this exercise, you should have configured the Order Details report to execute on demand and then viewed the report in Report Manager.

Exercise 3: Configuring and Viewing a Cached Report

Scenario

Adventure Works management has decided that the Product Sales YTD report takes too long to run, especially considering the number of times the report is run in a short time period. You need to help speed performance by allowing the report to be cached for use, and the data needs to be refreshed every 30 minutes.

In this exercise you will configure a report to execute as a cached instance that expires every 30 minutes.

The main tasks for this exercise are as follows:

1. Configure the Product Sales YTD report for cached execution.
2. Expire copies of the Product Sales YTD report on a schedule.

► Task 1: Configure the Product Sales YTD report for cached execution

- In Report Manager, go to the **Product Sales** folder then and click **Show Details**.
- On the **Execution** link of the properties for the **Product Sales YTD** report and click on **Cache a temporary copy of the report. Expire copy of report after a number of minutes**.
- Apply the default settings and note the warning message.
- Under the **Data Sources** link, go to **Product Sales**, and click the **AdventureWorksDW2008** link.
- Change the **Connect Using property to Credentials stored securely in the report server** and enter the login name of **NY-SQL-01/Student**, with the password of **Pa\$\$w0rd**.
- Verify that the **Use as Windows credentials when connecting to the data source** check box is selected, and that the **Impersonate the authenticated user after a connection has been made to the data source** check box is not selected.
- Apply the changes and go back to the **Execution** link.

- Set the report to **Cache a temporary copy of the report. Expire copy of report after a number of minutes** and apply the settings. Notice that there is no warning this time.
 - View the report to create the first cached instance.
- **Task 2: Expire copies of the Product Sales YTD report on a schedule**
- In the **Execution** link under **Properties** configure the cached expiration based on a schedule to occur daily at 11 P.M.
 - Apply the change and view the report to create the cached instance for the schedule.

Results: After this exercise, you should have set the cached instance of the Product Sales YTD report to expire every day at 11 P.M.

Exercise 4: Configuring and Viewing a Snapshot Report

Scenario

Accounting has a Reseller Sales report that needs to be run on the first of each month. However, sometimes the report is not run before new data is entered. You are tasked with making sure the report runs on the first of each month before anyone has entered data for the new month.

In this exercise, you will schedule Reseller Sales report to create a snapshot at the beginning of each month. After the snapshot is created you will view it from report history.

The main tasks for this exercise are as follows:

1. Change the Reseller Sales report to use AdventureWorksDW2008.
2. Schedule a snapshot to run at the beginning of the month.
3. Store Reseller Sales snapshots in report history.
4. View the Reseller Sales report and report history.

► **Task 1: Change the Reseller Sales report to use AdventureWorksDW2008**

- Edit the properties of the **Reseller Sales** report in **Show Details** link of the **Reseller Sales** folder from the **Home** page.
- Change the data source for the **Reseller Sales** report to use the **Product Sales\AdventureWorksDW2008** shared data source.
- Apply the changes.

► **Task 2: Schedule a snapshot to run at the beginning of the month**

- On the **Report Manager Site Settings** link, go to **Schedules** and create a new schedule.
- Set up a schedule named **Beginning of Month** to occur:
 - Monthly
 - On the first day of the month
 - At 5:00 A.M.

- Back in the **Reseller Sales** folder, edit the **Reseller Sales** execution properties as follows:
 - Render the report from an execution snapshot.
 - Use the shared schedule **Beginning of Month**.
 - Do not create a snapshot when the **Apply** button is clicked.
 - Apply the property changes.
- **Task 3: Store Reseller Sales in snapshots in report history**
- In Report Manager, go to **History** and:
 - Store all report execution snapshots in history.
 - Limit the number of snapshot copies to 12.
 - Apply the settings.
- **Task 4: View the Reseller Sales report and report history**
- View the **Reseller Sales** report and note the error message.
 - Under **Properties**, go to **Execution** and set the report to create a snapshot when the **Apply** button is pressed.
 - View the report.
 - Check the **History** link to verify that the snapshot was created.
 - Close Report Manager and shutdown the virtual machine discarding any changes.

Results: After this exercise, you should have created a snapshot history for the Reseller Sales report and viewed the report history.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips
- Real-world Issues and Scenarios
- Best Practices

Review Questions

1. Cached instances are similar in that they both store a copy of a report on the server. What is the one property that must be set on a cached instance that is not required by snapshots?
2. When configuring snapshots and cached instances special care must be given to the kind of authentication used in the data source. What is special about a data source's authentication settings?
3. When system or network resources are scarce during regular business hours, what can be done to make sure none of those resources are consumed by reporting?

Common Issues related to cached reports and report snapshots

Issue	Troubleshooting tip
You create a cached instance or a snapshot and the report does not run due to security issues	
A snapshot runs on a schedule but it does not show up in report history	

Real-world Issues and Scenarios

1. The Adventure Works accounting staff wants you to setup their month ending reports to run the fifth day of each month and keep a history of the reports for five years. What features and settings will you use to accomplish this?
2. The accounting department loves the scheduled reports and would like you to set the reports up to be sent in e-mail to the VP of Finance when they are run. How will you set the reports up to do this?
3. An organization has a large database for its enterprise resource planning (ERP) system that requires most of the data source server's resources and an excessive amount of time to create the required data sets for many of their daily reports. What can you do to make sure that the resources remain available for daily operations, yet the reports are quickly available for the users of the ERP system?
4. Your supervisor approaches you and says that the entire sales force of 50 is accessing a particular report several times a day. Although the report requires a minimal amount of processing time, the volume of use is placing a regularly heavy load on system resources. What can you do with the report that will make it available to of the users with the latest data, yet consume the least system resources?

Best Practices related to report caching and report snapshots

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

- You will typically want to view a report after setting up a schedule to create the first cached instance or snapshot in the schedule.
- When using stored credentials in a data source create a specialized user on the server that has security settings specific to running reports.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 7

Using Subscriptions to Distribute Reports

Contents:

Lesson 1: Introduction to Report Subscriptions	7-3
Lesson 2: Creating Report Subscriptions	7-8
Lesson 3: Managing Report Subscriptions	7-17
Lab: Implementing Subscriptions	7-24

Module Overview

- Introduction To Report Subscriptions
- Creating Report Subscriptions
- Managing Report Subscriptions

Subscriptions are standing requests with configurable parameters used to deliver reports to users. Subscription delivery can occur on regular or irregular intervals and reports delivered using subscriptions can be rendered in a variety of formats, including formats designed for both screen and paper viewing. Subscription delivery and security parameters are highly configurable.

The information in this module can be used to implement automation for reports. Automating report delivery can reduce administrative workload.

Lesson 1

Introduction to Report Subscriptions

- What Are Subscriptions?
- How Subscriptions Are Used
- Subscription Scenarios

Subscriptions are standing requests to deliver report data. Subscriptions are configurable in multiple ways, including the format of the report output, the delivery mechanism, and the delivery timing.

Because subscriptions can be configured once and deliver reports many times, administrative workload can be reduced.

What Are Subscriptions?

Defining Subscriptions

- Standing request to deliver a report at a specific time or in response to an event
- Application file format specified during subscription creation
- Alternative to running report on demand

Key Points

A subscription is a standing request to deliver a report at a specific time or in response to an event, and in an application file format that you specify in the subscription.

- A subscription is a standing request with configurable parameters.
- Time or response to an event, render format and delivery method are all configurable.
- Microsoft® SQL Server® Reporting Services supports two kinds of subscriptions: standard and data-driven.

Question: Describe some scenarios that might benefit from subscriptions.

How Subscriptions Are Used

Mechanisms to execute and deliver rendered reports

- Default delivery extensions:
 - E-mail (SMTP)
 - File share
 - SharePoint library
 - Custom

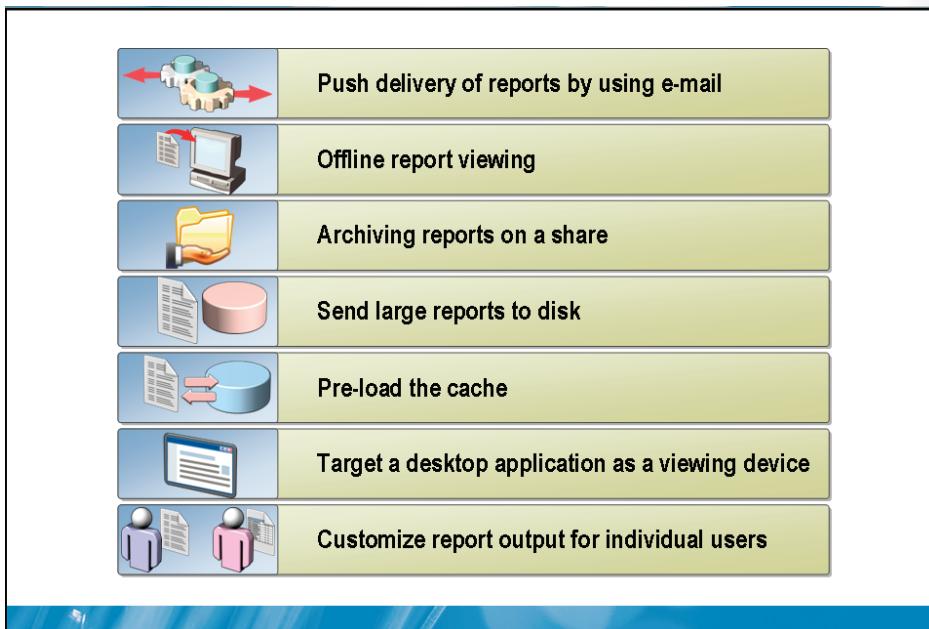
Key Points

Subscriptions are used to deliver rendered reports to end users at regular intervals or in response to events.

- Multiple delivery options are available to service a variety of scenarios.
 - Send reports to users in e-mail
 - Deliver to a file share
 - Deliver to a Microsoft Office SharePoint® server

Question: What criteria might be used in choosing a subscription delivery type?

Subscription Scenarios



Key Points

Because subscriptions are automated and flexible, they can provide tremendous efficiency gains across a variety of scenarios.

Subscriptions are useful in multiple scenarios.

- E-mail can be used to deliver reports to single users or lists of users.
- Adobe PDF, Web archive, or Microsoft Office Excel® formats can be used for offline report viewing.
- Reports can be archived to a network file share which can be backed up using normal backup procedures.
- Large reports that would take too long to display in a browser can be sent to disk instead.
- The cache can be pre-loaded to reduce waiting time for users.

- Reports can target a desktop application as a viewing device.
- A data-driven subscription can customize report output format, delivery options, and report parameter settings for individual users.

Question: Describe one or more scenarios in which subscriptions would not be the best tool for report delivery.

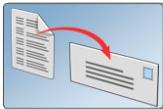
Lesson 2

Creating Report Subscriptions

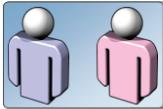
- What Are Standard Subscriptions
- Creating Standard Subscriptions
- Subscription Rendering Types
- What Are Data-Driven Subscriptions
- Creating Data-Driven Subscriptions

Subscriptions are either standard or data-driven subscriptions. Standard subscriptions are configured and then operate based on the initial configuration selected by either end-users or an administrator. Data-driven subscriptions are configured using data stored in a database table. Data-driven subscriptions are dynamic because updates to the configuration table will alter how the subscription delivers and formats data.

What Are Standard Subscriptions?



Delivers one rendered report to destination



Can be defined by end-user or administrator

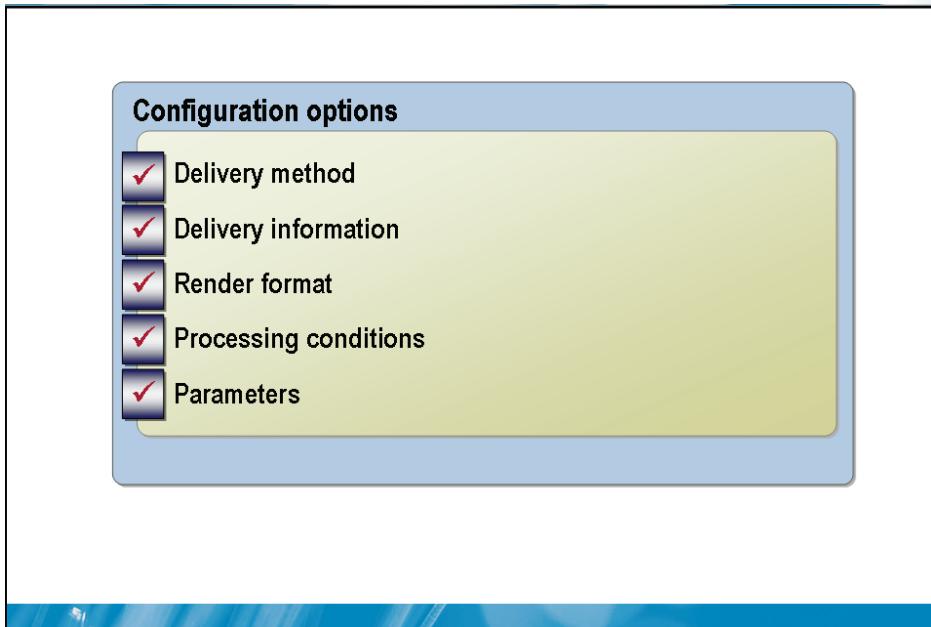
Key Points

A standard subscription delivers one rendered report to one destination, such as an email address (or addresses) or a single file share.

- Custom delivery extensions can be added.
- End-users and administrators can define subscriptions.

Question: In your organization, how might you implement allowing end-users to define subscriptions?

Creating Standard Subscriptions



Key Points

Use Reporting Services Configuration to configure delivery options, render format, and processing conditions.

- To use e-mail delivery, the report server must be configured for an SMTP server or gateway connection before you create the subscription.
- Check file share permissions if using the file share delivery method.
- Avoid rendering formats that do not render in a single stream.
- For a subscription based on a snapshot, you can execute the subscription upon update of the snapshot. For non-snapshot reports, you can create a schedule or use a shared schedule.

Question: What considerations are relevant when setting file share permissions for file share subscription delivery?

Subscription Rendering Types

Renderer Type	Format	Description
Soft page-break renderer	Excel	Opens a report in Microsoft Excel.
Soft page-break renderer	Word	Opens a report in Microsoft Word.
Soft page-break renderer	Web archive	Opens a report in MHTML. The report opens in Internet Explorer.
Hard page-break renderer	Acrobat (PDF) file	Opens a report in Adobe Acrobat Reader (version 6.0 or later).
Hard page-break renderer	TIFF	Opens a report in a page-oriented format.

Key Points

Export formats are supported through rendering extensions that are installed on the report server. Rendering refers to the process of exporting the report to another format, typically a file format, that can be used in other applications.

- Data renderers strip all formatting and layout information from the report and display only the data
- Soft page-break renderers maintain the report layout and formatting
- Soft page-break renderers are optimized for screen-based viewing and delivery

- Hard page-break renderers maintain the report layout and formatting
- Hard page-break renderers are optimized for a consistent printing experience, or to view the report online in a book format

Question: For each renderer type, describe one or two scenarios for which that renderer is the best choice.

Subscription Rendering Types (*continued*)

Renderer Type	Format	Description
Data renderer	XML	Opens a report in XML.
Data renderer	CSV	Opens a report in comma-delimited format.
Soft page-break renderer	HTML 4.0	If your browser support HTML 4.0, that is the format that is used. Otherwise, HTML 3.2 is used.
RPL stream	RPL Renderer	Opens a report in the Report Page Layout format.

Key Points

- RPL stream is a binary stream type.
- RPL stream preserves the report hierarchy.

Question: For each renderer type, describe one or two scenarios for which that renderer is the best choice.

What Are Data-Driven Subscriptions?

Deliver rendered reports by using dynamic values from a table

- Table fields include:
 - Dynamic list of destinations
 - Report rendering format
 - Parameter values

Key Points

A data-driven subscription can deliver a report in many rendered formats to many destinations.

- Data-driven subscriptions require a database table to store report values.
- Dynamic values used in a data-driven subscription are obtained when the subscription is processed.

Question: Describe some scenarios in which a data-driven subscription would be ideal.

Creating Data-Driven Subscriptions

- Configuration steps**
- 1** Create and populate a subscriber table
 - 2** Ensure that credentials are stored with the report
 - 3** Choose a delivery method
 - 4** Create a connection to the subscriber table
 - 5** Define a query to retrieve subscribers
 - 6** Set the subscription properties as for a standard subscription

Key Points

Data-driven subscriptions are typically created and maintained by report server administrators.

- The report must use stored credentials or no credentials to retrieve data at run time .
- You must have an accessible external data source that contains subscriber data.
- The author of the subscription must have permission to *Manage reports* and *Manage all subscriptions*.
- The author must also have the necessary credentials to access the external data source that contains subscriber data.

Question: Describe one or two scenarios in which data-driven subscriptions might be useful.

Demonstration: Creating Subscriptions

In this demonstration, you will see how to:

- Create a standard subscription (email subscription)
- Create a data-driven subscription

Question: Of what is a data-drive subscription composed?

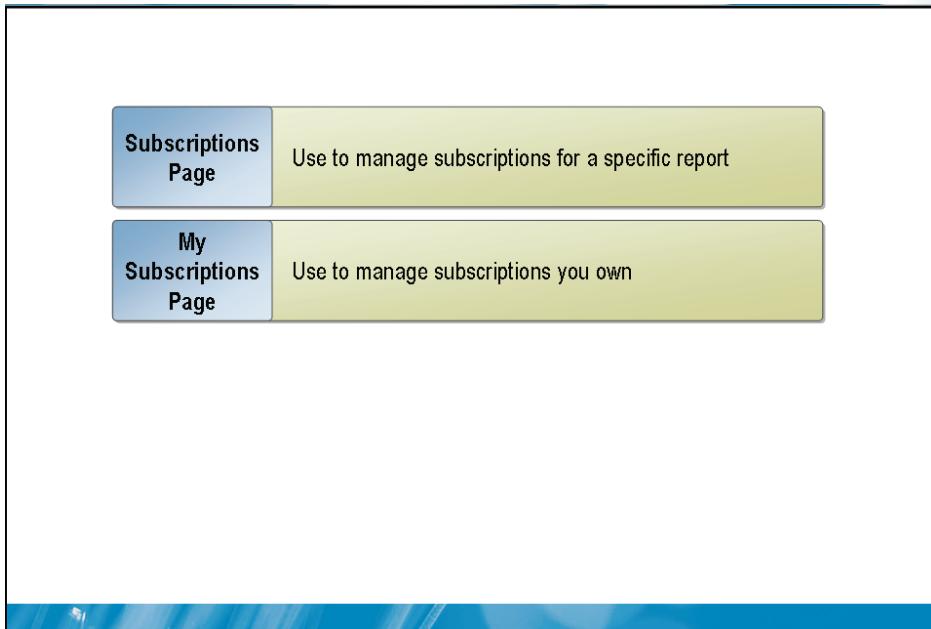
Lesson 3

Managing Report Subscriptions

- Managing Subscriptions with Report Manager
- Managing Subscription and Delivery Availability
- Controlling Report Distribution
- Monitoring Subscription Status

The Report Manager can be used to control and monitor subscriptions. Because the reports that are delivered using subscriptions can contain sensitive information, it is important to understand how Report Manager can be used to configure security for subscriptions.

Managing Subscriptions with Report Manager



Key Points

A user who creates a subscription owns that subscription.

- Create subscriptions on the **Subscription** page, not on the **My Subscriptions** page.
- Each user can modify or delete the subscriptions that he or she owns.
- The **My Subscriptions** page allows you to edit and delete existing subscriptions that you own.

Managing Subscription and Delivery Availability

Disabling Subscriptions

Remove the "Manage individual subscriptions" ability from the Browser role

Disabling Delivery Extensions

Edit the RSReportingServer configuration file and delete the appropriate delivery extension

Key Points

- Removing a delivery extension can result in inactive subscriptions.
- Removing subscription-related tasks prevents users from creating and modifying subscriptions.
- Managing security roles can keep the Reporting Services system controllable for administration staff.

Question: What are one or two business reasons for preventing users from creating or modifying subscriptions?

Controlling Report Distribution



Key Points

- Ensuring that users only receive appropriate reports that they are entitled to view requires security settings and control of distribution.
- Limit e-mail distribution to a list of specific host domains by modifying the RSReportServer configuration file.
- Sending the report server URL is the most effective way to control report distribution.

- To control file share delivery, use Access Control Lists (ACLs) to secure access to the shared folders where reports are stored.
- To use Windows® SharePoint® Services (WSS) security features with report server items, you must have a report server that runs in SharePoint integrated mode.

Question: How does sending the report server URL contribute to effective control of report distribution?

Demonstration: Controlling Report Distribution

In this demonstration, you will see how to:

- Modify RSReportServer configuration file to show the **To** field in a subscription

Question: Can all Reporting Services configuration files be modified with a text editor?

Monitoring Subscription Status

Status	Description
New subscription	Appears when you first create the subscription
Inactive	Appears when a subscription is inactive
Done: n processed of n_1 total; n_2 errors	Shows the status of a data-driven subscription execution including current statistics
Failure sending mail	The report server did not connect to the mail server
File x was written to y	The delivery to the file share location was successful
An unknown error occurred when writing file	The delivery to the file share location did not succeed
Failure connecting to the destination folder	The folder specified could not be found
The file x could not be written to y	The file could not be updated with a newer version
Failure writing file x	The delivery to the file share location did not succeed

Key Points

Individual users can monitor the status of a subscription using the **My Subscriptions** page or a **Subscriptions** tab in Report Manager.

- Report server administrators can review the reportserverservice_*.log files to determine subscription delivery status.
- Report server administrators can also monitor standard subscriptions that are currently processing.
- Data-driven subscriptions cannot be monitored.

Question: Describe two or three situations that could cause a **Failure writing file x** error.

Lab: Implementing Subscriptions

- Exercise 1: Creating a Standard Subscription
- Exercise 2: Creating a Data-Driven Subscription

Logon information

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

Estimated time: 60 minutes

Exercise 1: Creating a Standard Subscription

Scenario

The Adventure Works team has asked you to publish a number of pre-written reports to a report server. To do this you will create a standard subscription for the Product Profitability report. You will create a standard subscription that will deliver the report in Excel format to a specific file share.

The main tasks for this exercise are as follows:

1. Set up the lab environment.
2. Open and deploy the AdventureWorks solution.
3. Configure a shared data source.
4. Create a standard subscription.
5. View the standard subscription output.

- ▶ **Task 1: Set up the lab environment**
 - Launch 6236A-NY-SQL-01.
 - Log on as **Administrator** with the password of **Pa\$\$w0rd**.
 - Run E:\Allfiles\Mod07\Labfiles\Starter\Exercise01\runScripts.bat.
- ▶ **Task 2: Open and deploy the AdventureWorks solution**
 - Use SQL Server Business Intelligence Development Studio to open E:\Mod07\Allfiles\Labfiles\Starter\Exercise01\AdventureWorks\AdventureWorks.sln.
 - Use the Solution Explorer to deploy the **AdventureWorks** solution.
 - In the Output window, confirm that all three projects were successfully deployed without warnings.
- ▶ **Task 3: Configure a shared data source**
 - Connect to <http://NY-SQL-01/reports> as **Administrator** with a password of **Pa\$\$w0rd**.
 - Navigate to **Product Sales | AdventureWorksDW**.
 - Use the following settings:
 - Under **Connect Using**, select **credentials stored securely in the report server**.
 - In the **User name** field, type **NY-SQL-01\Administrator**.
 - In the **Password** field, type **Pa\$\$w0rd**.
 - Select the **Use as Windows credentials when connecting to the data source** check box.
 - Ensure that the **Impersonate the authenticated user after a connection has been made to the data source** check box is not selected.

► **Task 4: Create a standard subscription**

- Navigate to Product Sales | Show Details | Product Profitability Properties.
- Navigate to Subscriptions tab | New Subscriptions.
- Use the following settings:
 - Delivered by: **Windows File Share**
 - Add a file extension when the file is created: **selected**
 - Path text field: **\NY-SQL-01\Reports**
 - Render Format: **Excel**
 - Overwrite options: **Increment file names as newer versions are added selected**
 - Run the subscription: **When the scheduled report run is complete**
- Click **Select Schedule** and use the following settings:
 - Select the **Once** option
 - Set the **Start time** to be three minutes later than the current time
 - **Year** and **Month** parameters: **Use Default**
 - Category: **All Product**
 - User Name: **NY-SQL-01\Administrator**
 - Password: **Pa\$\$w0rd**

► **Task 5: View the standard subscription output**

- View the contents of **\NY-SQL-01\Reports\ Product Profitability.xls**.

Results: After this exercise, you should have created and viewed the output of a standard subscription.

Exercise 2: Creating a Data-Driven Subscription

Scenario

In this exercise, you will create a data-driven subscription to deliver the Actual Vs Quota report to the company's sales directors.

The main tasks for this exercise are as follows:

1. Change the Actual Vs Quota report to use a shared data source.
2. View the SubscriptionGroupDirector table.
3. Create a data-driven subscription.
4. View the subscription notifications.

► **Task 1: Change the Actual Vs Quota report to use a shared data source**

- On the **Home** page in Report Manager, click the **Territory Sales** folder link, and then click **Actual Vs Quota report**.
- Navigate to **Properties tab | Data Sources**.
- Set the **A shared data source option** to **Product Sales \AdventureWorksDW2008**.

► **Task 2: View the SubscriptionGroupDirector table**

- Start SQL Server Management Studio.
- Use the following parameters:
 - Server type: **Database Engine**
 - Server name: **NY-SQL-01**
- View the Top 1000 rows in the **SubscriptionGroupDirector** table.

► **Task 3: Create a data-driven subscription**

- In Report Manager navigate, to **Subscriptions** tab for the **Actual Vs Quota report** | **New Data-driven Subscription**.
- Use the following settings:
 - Description: **Actual Vs Quota Subscription**
 - Specify how recipients are notified: **E-mail**
 - Select **Specify a shared data source** and then select **Product Sales\AdventureWorksDW2008**.
- Query: **SELECT*FROM dbo.SubscriptionGroupDirector**



Note: Be sure to click **Validate** to verify that a **Query validated successfully** message is displayed.

- Configure the properties by using the following settings:

Property	Option	Value
To	Get the value from the database	To
Cc	No value	None
Bcc	No value	None
Reply-To	Specify a static value	Sales@adventure-works.com
Include Report	Get the value from the database	IncludeReport
Render Format	Get the value from the database	RenderFormat
Priority	Specify a static value	Normal
Subject	Specify a static value	@ReportName was executed at @ExecutionTime
Comment	No value	None
Include Link	Get the value from the database	IncludeLink

- Complete the wizard using the following settings:
 - CalendarYear: **Use Default**
 - Group: **Get the value from the database**, and in the drop-down list, click **GroupParameter**
 - Select **On a schedule created for this subscription**
 - Schedule details: **Once**
 - Set the start time to execute two minutes later than the current time.
- **Task 4: View the subscription notifications**
- Wait until the time you scheduled the subscription to run, and then navigate to **C:\inetpub\mailroot\Drop**.
 - Examine the contents of the **.eml** files in the **Drop** folder.
 - Turn off **6236A-NY-SQL-01** and delete changes.

Results: After this exercise, you should have successfully created and viewed the output of a data-driven subscription.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips

Review Questions

1. A subscription is created as an alternative to what?
2. What are the default delivery extensions?
3. What dynamic values are used for data-driven subscriptions?
4. What four options are used to control report distribution?

Common Issues and Troubleshooting Tips

Issue	Troubleshooting tip
Unable to Send Reports Using E-Mail With Windows Server 2003 and POP3.	
Failure sending mail: The server rejected the sender address. The server response was: 454 5.7.3 Client does not have permission to submit mail to this server.	
Subscriptions are not processing.	

MCT USE ONLY. STUDENT USE PROHIBITED

Module 8

Administering Reporting Services

Contents:

Lesson 1: Reporting Services Administration	8-3
Lesson 2: Performance and Reliability Monitoring	8-11
Lesson 3: Administering Report Server Databases	8-23
Lesson 4: Security Administration	8-32
Lesson 5: Upgrading to Report Services 2008	8-46
Lab: Administering Reporting Services	8-49

Module Overview

- Reporting Server Administration
- Performance and Reliability Monitoring
- Administering Report Server Databases
- Security Administration
- Upgrading to Report Services 2008

To facilitate administration, Microsoft® SQL Server® 2008 Reporting Services (SSRS) includes configuration files that can be modified to adjust data connection strings, SMTP server settings, set code access security policies, and set report designer extensions. In this module, students will learn how to administer the Reporting Services server, how to monitor and optimize the performance of the report server, how to maintain the Reporting Services databases, and how to keep the system secure.

A Reporting Services installation includes several tools that can be used to administer and manage a Reporting Services deployment including Report Manager, the Reporting Services Configuration Tool command line tools like the rs, rsconfig, and rskeymgmt Utilities.

Lesson 1

Reporting Services Administration

- Reporting Services Configuration Files
- Reporting Services Configuration Manager
- Assigning Service Accounts

Reporting Services includes configuration files that can be modified to adjust data connection strings, SMTP server settings, set code access security policies, and set report designer extensions. Reporting Services supports performance and reliability monitoring and security administration. Reporting Services can be integrated with SharePoint® servers using SharePoint Integrated Server.

In this lesson, you will learn how to administer Reporting Services components, such as the Reporting Services configuration files and Reporting Services security. You will also look at the new features in the Report Server architecture and look at the Reporting Services configuration manager.

Reporting Services Configuration Files

Configuration File	Settings
RSReportServer	Data connection strings, SMTP server settings, Delivery extensions, rendering and data source extensions
ReportingServicesService	Trace level and log files
RSMgrPolicy	Code access security policies for Report Manager
RSSrvPolicy	Code access security policies
RSReportDesigner	Rendering, data source, and designer extensions
RSPreviewPolicy	Access security policies for the server extensions used during report preview

A Reporting Services installation includes several Extensible Markup Language (XML) configuration files that control the behavior of various Reporting Services components. After installation, you may need to modify these configuration files to add or remove particular features, using the configuration tool or any XML editor.

Key Points

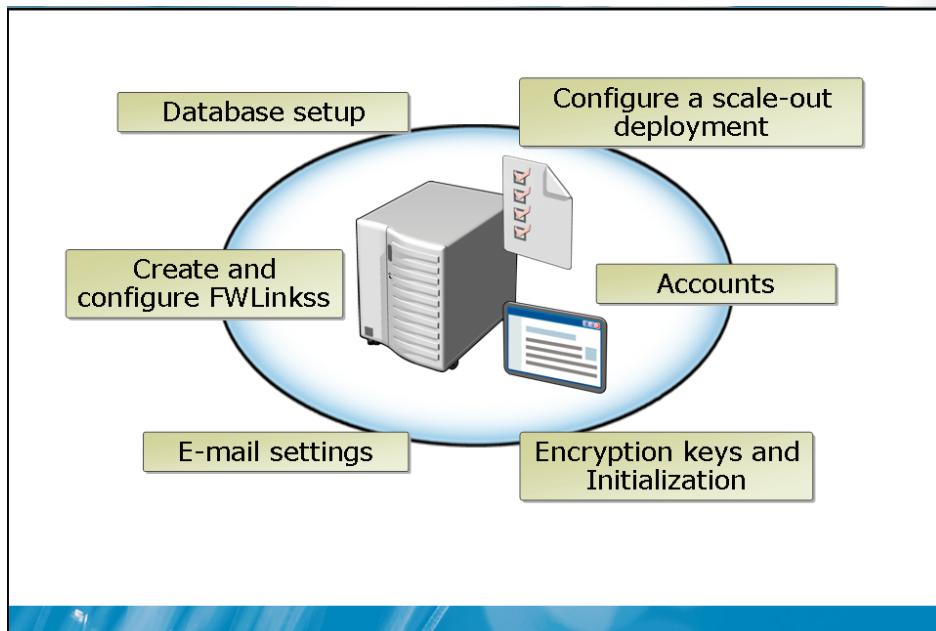
- RSReportServer.config is located in the Microsoft SQL Server \<SQL Server Instance>\Reporting Services\ReportServer folder, and it stores configuration settings for feature areas of the Report Server service: Report Manager, the Report Server Web service, and background processing.
- RSSrvPolicy.config is located in the Microsoft SQL Server \<SQL Server Instance>\Reporting Services\ReportServer folder, and it stores the code access security policies for the server extensions.
- The RSMgrPolicy.config configuration file is located in Microsoft SQL Server \<SQL Server Instance>\Reporting Services\ReportManager, and it stores the code access security policies for Report Manager.

- ReportingServicesService.exe.config is located in Microsoft SQL Server \<SQL Server Instance>\Reporting Services\ReportServer\bin and stores configuration settings that specify the trace levels and logging options for the Report Server service.
- RSReportDesigner.config is located in the \Program Files\Visual Studio 9.0 \Common7\IDE\PrivateAssemblies folder, and it contains settings for Report Designer. You can configure the options for this file as shown in the following table.
- RSPreviewPolicy.config is located in <drive>:\Program Files \Microsoft SQL Server\100\Tools \ReportDesigner and stores the code access security policies for the server extensions used during report preview.

You can create or add extension classes for all three elements by adding them to the appropriate element within the configuration file.

Question: What advantage is there in putting settings for various components in separate configuration files instead of a single file?

Reporting Services Configuration Manager



The Reporting Services Configuration Manager provides a graphical user interface to modify several configuration settings stored in the previously described configuration files. These settings are grouped into the following core areas.

Key Points

- **Server status.** Provides information about the status of the Report Server Microsoft Windows® service (running or not), and provides buttons to start and stop the service.
- **Virtual directories.** Enables configuration of the Report Server Web service and the Report Manager Web site virtual directories. This allows you to change the virtual directory names after installation, as well as requiring Secure Sockets Layer (SSL) connections for the Web service if necessary.
- **Service accounts.** Enables configuration of the three Windows accounts that Reporting Services uses.
- **Report Server database setup.** Creates and configures the report server databases that provide internal storage for one or more report server instances.

- **Encryption keys and initialization.** Manages the symmetric key that is used to encrypt and decrypt data in a report server. This includes creating backups of encryption keys so that you can change service accounts, migrating to a different server, or initializing a new report server instance to share the report server database.
- **Server e-mail.** You can use the Reporting Services Configuration tool to specify which SMTP server or gateway on your network to use for e-mail delivery.



E-mail delivery. Configures the Simple Mail Transport Protocol (SMTP) settings that enable the Report Server e-mail delivery extension.

Question: How would you change the Reporting Services database credentials using Configuration Manager for a scale-out deployment?

Assigning Service Accounts

Account	Options
Report Server Windows service	<ul style="list-style-type: none">● Least-privileged Windows user account● Network service account (recommended for Windows Server 2008)● Local System account (too many permissions)● Local Service account (too few permissions)
Report Server Database Connection	<ul style="list-style-type: none">● Windows authentication● SQL authentication
Unattended Execution	<ul style="list-style-type: none">● Windows domain account● Windows local account
SharePoint Integrated Server	<ul style="list-style-type: none">● SharePoint permissions must be updated to reflect the Reporting Services service account

Some Reporting Services components require specific user credentials in order to function. You will need to modify these settings after installation if you plan to change the user accounts used by Reporting Services.

Key Points

- The Report Server Windows service can run as any of the following account options:
 - **Network Service account.** This option is only available in Microsoft Windows Server® 2003 or later and is the recommended option.
 - **Least-privileged Windows user account.** Ensure that the Microsoft Windows® account can make network connections.
 - **Local System account.** This option includes more permissions than are necessary and is therefore not recommended.
 - **Local Service account.** This option will not work in most scenarios. Avoid using this built-in account unless all client and server operations are performed exclusively on the local computer.

- The Report Server Database Connection can use Windows authentication or SQL authentication.
- You can use a Windows domain account or a Windows local account for unattended execution.
- Finally, when using SharePoint® Integrated Mode, SharePoint permissions must be updated to reflect the Reporting Services service account.

You can also set this account using the Windows Services tool as you would for any other Windows service.

Question: When would you want to use a domain account over a network service account?

Demonstration: Using Reporting Services Configuration Manager

In this demonstration, you will see how to:

- View Reporting Services settings
- Back up the Reporting Services encryption keys

Question: How can you recover encrypted data if you need to restore your encryption keys but don't have a backup?

Lesson 2

Performance and Reliability Monitoring

- Using Trace Files
- Logging Report Execution
- Using Performance Counters
- Applying Timeouts
- Managing Processes

To assist with troubleshooting and performance monitoring, Reporting Services supports trace logs which can record error information and log report server requests. Likewise, performance counters can log report activity such as how many reports were executed per second, the total requests made for a given report and metrics around how the web services is functioning. To troubleshoot performance problems, you can set timeouts on queries to ensure a failed query does not monopolize CPU cycles.

In this lesson, you will learn how to use trace files and log execution reporting, apply timeouts, use timeouts and suspend jobs to troubleshoot or monitor Reporting Services activity.

Using Trace Files

Trace file allow you to monitor the report server status

- Trace file:
 - Confirm report delivery and Record error information
 - Monitor report execution activity
 - Location: \Microsoft SQL Server\<SQL Server Instance>\Reporting Services\LogFiles\ReportServerService_<timestamp>.log
- End-to-end logging
 - Records all HTTP requests handled by report server
- New trace file created daily, old files are not removed
- Control level of tracing using DefaultTraceSwitch

Reporting Services provides trace files with which you can monitor the status of the report server. Becoming familiar with these trace files will help you assess reasons for reporting problems if they should occur.

Key Points

- Trace logs contain information about various report server operations, including the following:
 - System information
 - Events logged in the Application log
 - Exceptions and warnings generated by the report server
 - Inbound and outbound SOAP envelopes
 - HTTP header, stack trace, and debug trace information
- You can review trace logs to confirm report delivery, monitor report execution activity, and view error information.

It is recommended that you do not disable tracing completely, so that you can at least log exception information.

Question: Under what circumstances might you want to turn off a trace or trace just exceptions and restarts?

Logging Report Execution

The report execution log provides valuable information

- Report execution logs to the Report Server database, and allows:
 - Monitoring of execution performance over time
 - Viewing frequency of report requests and the users who request them
- Logs information including:
 - Data retrieval, processing and rendering times, and report source
- Use supplied SQL Server® Information Services package to browse log information

One of the most important aspects of Reporting Services administration is logging report execution. This type of log information offers troubleshooting and monitoring benefits particularly related to report execution.

Key Points

- Report execution information is logged directly in the report server database. The log information allows you to view the frequency of particular report requests, as well as which users make them.
- For each report execution, there is a variety of information captured, including:
 - **Data retrieval time.** Time (in milliseconds) spent executing the query.
 - **Processing time.** Time (in milliseconds) spent processing the report, including applying filters, subtotals, grouping, sorting, and so on.
 - **Rendering time.** Time (in milliseconds) spent rendering the report in a specific format.
 - **Source.** Origin of the report execution.

- Reporting Services stores the log data in the **ExecutionLog** table in the report server database.
- You can also use the sample to export records from the execution log on a periodic basis by creating a schedule.

Question: How could you use execution logging to determine the most popular report in your environment and use execution logging data to improve the performance of that report?

Using Performance Counters

- Performance counters provide statistical information about Reporting Services applications

Service	Commonly Used Counters
ReportServer:Service Performance Object	<ul style="list-style-type: none">• Active Connections• Reports Executed per second• Total Cache Hits• Total Requests
Web Service	<ul style="list-style-type: none">• Deliveries/Sec• Total Processing Failures

When reports are executed, statistical information is created in the form of performance counters. This information is viewable through the Performance tool, Performance Logs and Alerts, and Task Manager.

Key Points

- The Reporting Services Web Service performance object includes 18 performance counters that relate to the performance of the report server.
- These counters are categorized as counters that show current activity or counters that show a total number, counting everything since the Web service was last started. Deciding which counters you should use will depend on your specific needs.
- The Reporting Services ReportServer:Service Performance Object includes 21 performance counters that relate to the performance of scheduling and delivery.

- This includes subscription and delivery, report execution snapshots, and report history statistics.
- These counters are categorized in the same way as the Web service performance counters.

Question: How would you use performance counter data to evaluate the performance of your Report Server web service?

Applying Timeouts

Timeouts prevent long running report issues

- Source query timeouts
 - Apply to the query execution time
 - Configured per data set query
 - Return a failure when timeout is exceeded
- Report execution timeouts
 - Apply to the total report execution time
 - Configured globally or per report
 - Return a failure when timeout is exceeded

Occasionally, a report may take a long time to execute. If reports take too long to execute, users can become impatient, which can often lead them to trying to run the report again, before the original request has been satisfied. This in turn can slow down the system further. You can configure timeouts in order to prevent such problems from occurring.

Key Points

- Source query timeouts help you prevent unpredictable source queries from running for an unreasonable length of time preventing the queries from monopolizing system resources.
- You can specify the maximum time limit for individual data set queries during report design by using the **Dataset** dialog box in Report Designer forcing a failure with user feedback.
- Report execution timeouts help you prevent any long-running report executions that go beyond the acceptable threshold.

- You can specify the maximum time limit for overall report execution in Site Settings in Report Manager.
- Alternatively, you can configure a timeout for each report individually in the report execution properties. When the timeout is exceeded, the user receives an error message in the browser window.



Tip: You can also cancel long-running queries or subscriptions manually by using the **Manage Jobs** page in Report Manager.

Question: What is a potential danger with using report timeouts?

Managing Processes

Managing Running Processes

- Viewing and Canceling Jobs:
 - On-demand report processing
 - Scheduled report processing
 - Standard subscriptions owned by individual users
- Pausing Report and Subscription Processing:
 - Modify Role Assignments to prevent access
 - Disable a shared data source
 - Pause a shared schedule

SQL Server Reporting Services monitors the status of all jobs that are running on the report server and include jobs where a query is being executed and where a report is being processed or executed. Being able to cancel a job or pause a report can be helpful if a computer needs to be taken offline or a running job that is taking too long to process needs to be stopped.

Key Points

- **Viewing and Canceling Jobs:** Canceling a job only cancels the processes that are running on the report server.
 - Because the report server does not manage data processing that occurs on other computers, you must manually cancel query processes that are subsequently orphaned on other systems.
 - Running jobs can be viewed through Management Studio in the Jobs folder.

- **Pausing Report and Subscription Processing:** Administrators cannot pause a report or subscription directly. However they can interrupt a report processing prior to the process starting or when a data source connection is made.

Question: When would you cancel a job rather than pause it?

Demonstration: Using Performance and Reliability Tools

In this demonstration, you will see how to:

- Add a new Reporting Service performance counter
- Create a new Reporting Service Data Collector Set

Question: What are some of the benefits of monitoring specific performance counters?

Lesson 3

Administering Report Server Databases

- Management Tools for Reporting Services
- Reporting Services Database Storage
- Defining a Backup and Restore Strategy
- Determining Disk Space Requirements for Reporting Services

A Reporting Services deployment uses two SQL Server relational databases for internal storage. By default, the databases are named **ReportServer** and **ReportServerTempdb**. **ReportServerTempdb** is created with the primary report server database and is used to store temporary data, session information, and cached reports. In Reporting Services, database administration tasks include backing up and restoring the report server databases and managing the encryption keys that are used to encrypt and decrypt sensitive data.

In this lesson, you will learn how to anticipate and estimate Reporting Services database space needs and define and implement a backup and restore strategy. You will learn how to backup and restore Reporting Services encryption keys as well as databases.

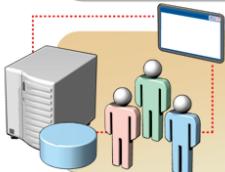
Management Tools for Reporting Services



Reporting Services Configuration Manager. Use the Reporting Services Configuration tool to configure a Reporting Services installation.



Report Manager. Report Manager is a Web-based report access and management tool that you use to administer a single report server instance from a remote location over an HTTP connection.



SQL Server Management Studio. SQL Server Management Studio is an integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server.

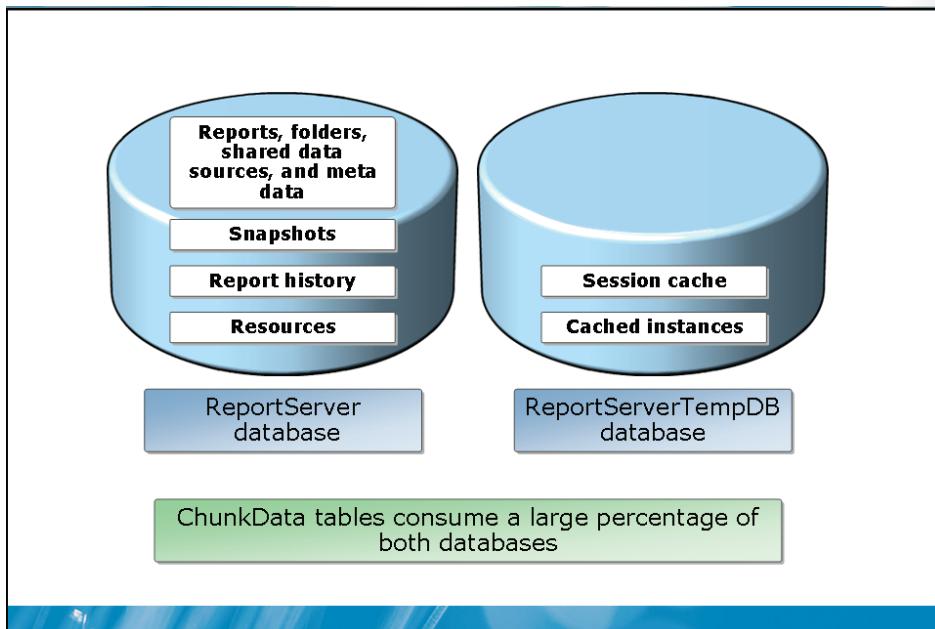
Key Points

Reporting Services for SQL Server 2008 comes with powerful management tools that can streamline report development and deployment.

- Reporting Services Configuration Manager. This tool is used to configure a Reporting Services installation. Reporting Services can be reconfigured after installation to support changing business requirements and scenarios.
- Report Manager. Report Manager is a Web-based report access and management tool that you access through Microsoft Internet Explorer 6.0 or later.
- SQL Server Management Studio. Through Management Studio, you can set up features like FILESTREAM storage, manage report jobs, and manage security.

Question: Which tool would be used by a content manager to find, run, and subscribe to published reports?

Reporting Services Database Storage



Reporting Services requires two SQL Server databases in order to operate: the **ReportServer** database and the **ReportServerTempDB** database. Both databases are created automatically during installation. Understanding what information is stored in each database will help you manage the physical files that make up the databases.

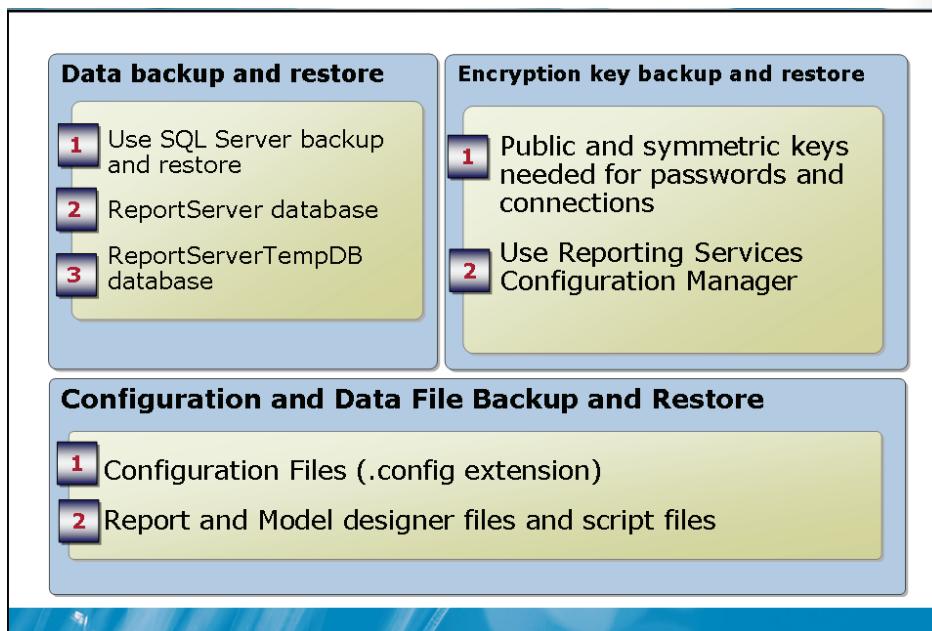
Key Points

- The following items are stored in the **ReportServer** database:
 - Reports, folders, shared data sources, and meta data (includes the actual report and data source definitions)
 - Resources (includes images associated with reports)
 - Snapshots (includes the current data for snapshot reports)
 - Report history (includes previous snapshot reports)

- The following items are stored in the **ReportServerTempDB** database:
 - Session cache (a user-specific report cache)
 - Cached instances (the report cache for multiple users)
- The **ChunkData** table exists mainly for backward compatibility with the SQL Server 2005 catalog.
 - It may contain large amounts of data if your SQL Server 2008 database was upgraded from SQL Server 2005.
 - SQL Server 2008 uses a new method for storing chunks. It shreds chunks across multiple rows and stores the data in the **Segment** table.

Question: How might the fact that the temporary database stores session data affect a backup strategy?

Defining a Backup and Restore Strategy



You will need to implement a backup and restore strategy for your report server databases. Additionally, you must back up the key used to encrypt content stored within each database. This will help you recover from any unexpected failures.

Key Points

- Because **ReportServer** and **ReportServerTempDB** are SQL Server databases, the primary mechanism for backup and restore is SQL Server backup and restore.
- Backup is considered mandatory for the **ReportServer** database because it contains the report and data source definitions.

- Each report server database encrypts stored user password and connection information.
- To manage the cryptographic key, use the Reporting Services Configuration Management tool or the encryption key management command line utility, rskeymgmt.exe.

Question: Name some important considerations when developing a backup strategy?

Determining Disk Space Requirements for Reporting Services

Steps to estimate database sizes

- 1** Estimate total number of reports
- 2** Examine intermediate report size
- 3** Factor in intermediate report persistence for ReportServer database
- 4** Factor in caching for ReportServerTempDB database

Before you install Reporting Services, you will need to estimate the size of your report server databases so that you can plan any additional hardware requirements.

Key Points

- To determine disk space requirements for your **ReportServer** and **ReportServerTempDB** databases, consider the following general estimation guidelines:
 1. Estimate the total number of reports.
 2. Examine the intermediate report size.

3. For the **ReportServer** database, you need to factor in the persistence of the intermediate reports.
4. For the **ReportServerTempDB** database, you need to factor in session caching.

Question: What impact does session data and snapshot data have on estimating database storage size and what are some ways to mitigate estimation difficulties?

Demonstration: Back Up and Restore Reporting Service Objects

In this demonstration, you will see how to:

- Backup an encryption key with Reporting Services Configuration Tool
- Restore an encryption key with Reporting Services Configuration Tool
- Backup a Reporting Services database with SQL Server Management Studio
- Restore a Reporting Services database with SQL Server Management Studio

Question: Under what circumstances might you need to backup and restore a Reporting Services installation?

Lesson 4

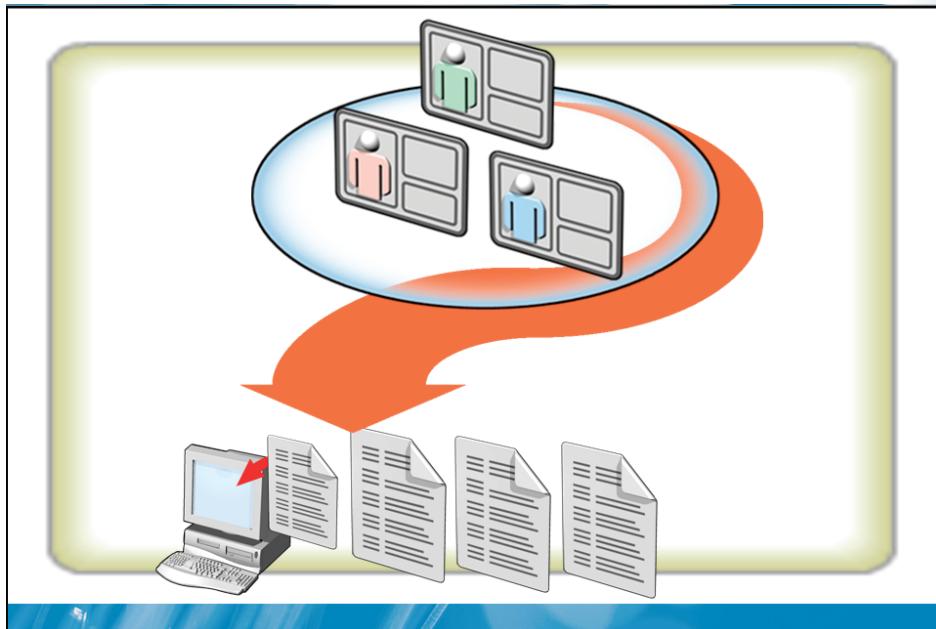
Security Administration

- The Reporting Services Authorization Model
- Assigning Roles
- Working with Item Role Definitions
- Securing Items
- Working with System-Level Role Definitions
- Securing the System

Reporting Services provides an authentication subsystem and role-based authorization model. Authentication and authorization models vary depending on whether the report server runs in native mode or SharePoint integrated mode. If the report server is part of a larger deployment of a SharePoint product or technology, SharePoint permissions determine who has access to the report server. Role-based security can simplify administration of rights to Reporting Services. In native mode, authorization is determined by Windows and SQL Server integrated security. In SharePoint integrated mode, security is based on permissions set in SharePoint.

In this lesson, you'll learn how to create and assign roles to Reporting Service objects, and secure individual items as well as set permissions at the system level.

The Reporting Services Authorization Model



Reporting Services uses authorization to ensure that only appropriate users can view individual reports or author new reports. You will also need to limit access to various features of the Report Manager, such as maintaining shared schedules.

Key Points

- The role-based security model supplied by Reporting Services is similar to that of other role-based security models used in other technologies, such as .NET Framework development and SQL Server.
- Report Manager allows you to administer a list of user permissions by linking a user or Windows group to a role.
- This approach provides a highly flexible and scalable security framework that enables you to manage the membership of a role instead of managing the security of each user individually.

- Reporting Services relies on the authentication capabilities of the underlying network.
- By default, Reporting Services integrates with Windows authentication.

Question: What is the advantage of using groups to assign permissions?

Assigning Roles

Assign roles with SQL Server Management Studio or Report Manager

- Base roles on tasks that users can perform
 - Tasks are predefined within the system and categorized as either item or system
- Assignment consists of three components:
 - Windows user account or group
 - Role definition — collection of item or system tasks
 - Securable object — item or system-level object

In Reporting Services, you administer security by creating role assignments in Report Manager. Assigning a role involves choosing appropriate tasks for that role and linking the role to users or groups and securable objects.

Both SQL Server Management Studio and Report Manager provide the ability to modify roles, add new roles, and assign the roles to users or groups and securable objects. The remaining topics in this lesson will describe how to set these security options.

Key Points

- A *role assignment* is a security policy that defines the tasks that users or groups can perform on specific items or branches of the report server folder hierarchy. Reporting Services provides predefined roles that you can use; you can also create your own roles.
- Reporting Services also defines a list of predefined tasks that you cannot add to or modify. These mutually exclusive tasks cover the full range of user and administrator actions, such as viewing reports, managing reports, and managing report server properties.

- Assigning Roles in SharePoint Integrated Mode. Windows SharePoint Services 3.0 provides built-in security features that you can use to grant access to report server items that you access from SharePoint sites and libraries. Security is configured through the integration settings between Windows SharePoint Services and a report server.

Question: How is a role assignment different from a permission setting?

Working with Item Role Definitions

- New item-level roles can be added
- Predefined item-level roles can be modified

Predefined Role	Description
Browser	View reports, resources, and folders
My Reports	Manage own My Reports folders
Publisher	Add content to the report server database
Content Manager	Deploy reports, manage data source connections, determine how reports are used

A *role definition* is a named collection of tasks that defines the type of access available to a user. Working with roles simplifies administration when compared to setting permissions on a user-by-user basis.

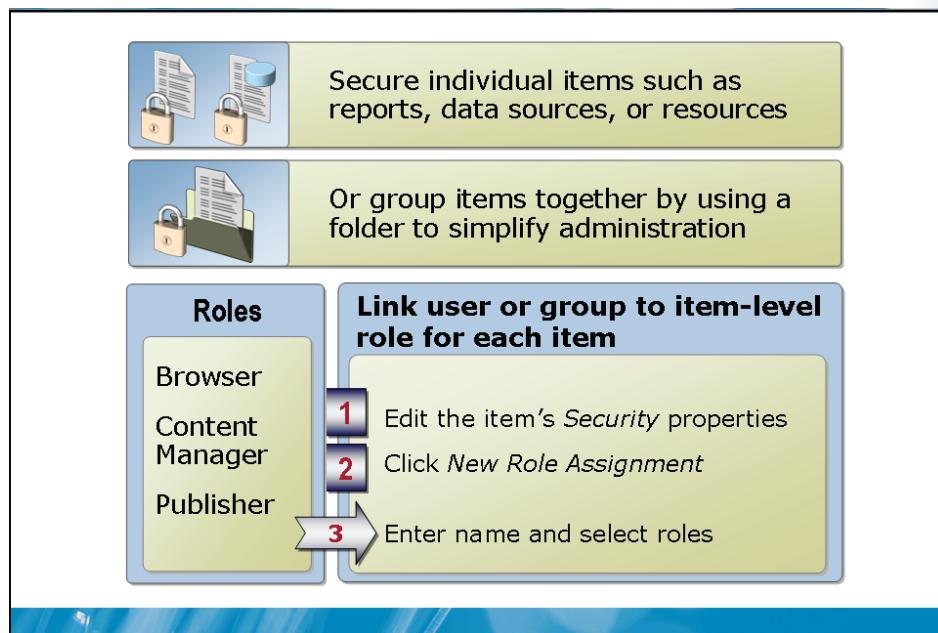
Key Points

- You can create your own roles if none of the predefined roles provides you with your required task grouping. Report Manager allows you to create item-level roles using the following steps:
 1. Navigate to the Site Settings page.
 2. Click **Configure item-level role definitions**, and then click **New Role**.
 3. Enter a role name and description in the **Name** and **Description** text boxes.
 4. Select the appropriate tasks from the list, and then click **OK**.
- You can then assign this new role to a user or group by using the **New Role Assignment** page within the individual item that needs to be secured.

- Reporting Services provides four predefined item-level roles that you can modify as appropriate by using Report Manager. For most roles, SharePoint includes an analogue when using SharePoint in integrated mode.
 - Browser (use Visitors group in SharePoint Integrated mode)
 - My Reports (There is no equivalent group. My Reports is not supported for a report server that runs in SharePoint integrated mode)
 - Publisher (use Members group in SharePoint Integrated mode)
 - Content Manager (use Owners group in SharePoint Integrated mode)

Question: When might you need to set permissions for a specific user rather than assigning a user to a role?

Securing Items



When you assign roles, you must specify which securable item you intend to grant access to. You can choose to secure items individually or as a group.

Key Points

- Individual items that you can secure include:
 - **Reports.** You can control whether a report is visible to users as well as whether users can change the report properties. Note that you can only secure snapshots and report history through their parent reports.
 - **Data sources.** You can secure shared data sources to limit which users can modify the data source settings.
 - **Resources.** Only standalone resources can be secured as individual items. Note that you cannot secure embedded resources separately from a report.
- Grouping items together into folders provides a simpler mechanism for security, because you do not have to secure several individual items.

- The Home folder is the root node of the folder hierarchy.
- In SharePoint integrated mode you can secure sites, lists, libraries, folders, and documents. A permission level is a set of permissions that can be granted to users or SharePoint groups on securable objects.

Question: Under what circumstances would you need to set the permissions for an individual item?

Working with System-Level Role Definitions

- New system-level roles can be added
- Predefined system-level roles can be modified

Predefined Role	Description
System User	View basic information about the report server
System Administrator	Administer report server but not content

For system-level security, roles affect the overall security of the report server site, not just individual items. You can administer these permissions by using the predefined system-level roles or by creating your own roles.

Key Points

- You can create new system-level roles by copying one of the predefined roles or by creating a new role.
- You can then assign this new role to a user or group using the System Role Assignments page.
- Reporting Services supplies two predefined system roles that you can modify: System User and System Administrator.

- The System User and System Administrator roles are not necessary for a report server that runs in SharePoint integrated mode. The report server does not provide any functionality that requires authorization at that level.

Question: What is the difference between system-level permissions and item-level permissions?

Securing the System



Report server site itself is the securable object



Assign security to users or groups

Link user or group to system-level role

- 1 In Site Settings, click *Configure site-wide security*
- 2 Click *New Role Assignment*
- 3 Enter user or group name and select roles

You can secure the report server site through system role assignments. This enables you to determine who has the ability to administer roles, manage jobs, set site properties, and so on.

Key Points

- The securable object for system security is the report server site, and its settings affect the site globally.
- Reporting Services does not create its own user accounts. Instead, it references existing local or domain accounts and groups defined in the operating system.
- If you inadvertently set role assignments in a way that locks all users out of the system, a local administrator can always reset security.
- Note that having access to a report server is not the same as having full access to all the reports and data it contains.

- In SharePoint integrated mode, the report server uses the authentication provider and permissions defined in the SharePoint Web application to control access. Permission to access items and operations is granted through SharePoint security policies that map a user or group account with a permission level, relative to an item.
 - In Reporting Services, both the Web service and Windows service require access to SharePoint databases.
 - The connection is managed internally.

Question: What two objects require a user to be assigned a system role in order to be accessed?

Demonstration: Modifying Role Assignments

In this demonstration, you will see how to:

- Modify an item-level role assignment
- Modify a system-level role assignment

Question: When granting access to report server items and operations, what steps should be followed to ensure permissions are applied properly?

Lesson 5

Upgrading to Report Services 2008

- Considerations for Upgrading
- Post Upgrade Tasks

Upgrade is performed by SQL Server Setup. SQL Server Setup can be used to upgrade any or all SQL Server components, including Reporting Services. To upgrade an earlier version of SQL Server, run the SQL Server 2008 Setup program on a computer that has an earlier version of SQL Server installed. Setup detects the existing instances and prompts you to upgrade. There are some items you'll need to consider before upgrading and some post-upgrade tasks of which you'll need to be aware.

In this lesson you'll learn of some of the important considerations you'll need to make before upgrading. You'll learn also how to perform some of the post-upgrade tasks.

Considerations for Upgrading

Item	Consideration
Component and Instances	Select components and instances or allow Setup to install SQL Server 2008 Reporting Services side-by-side existing installations
Editions and Versions	Upgrade support provided for <ul style="list-style-type: none">• SQL Server 2000 Reporting Services with SP2• SQL Server 2005 Reporting Services
Known Upgrade Issues	Review Readme for latest information about upgrade issues
Pre-Upgrade Checklist	Review Requirements for: <ul style="list-style-type: none">• SQL 2008 hardware and software• Parameters for System Configuration Checker• Security considerations for a SQL Server install

It's possible to upgrade an existing installation of Reporting Services or install a new instance of Reporting Services that runs side-by-side existing installations. To successfully perform an upgrade of Reporting Services, a number of factors need to be considered. Not all upgrade scenarios are supported and even on a successful upgrade, there are some post-install steps that will manually have to be performed.

Key Points

- Upgrades can only be done against SQL Server 2000 Reporting Services SP2 and SQL Server 2005 Reporting Services.
- MSDN publishes a pre-upgrade checklist and an Upgrade Advisor tool that will help surface any issues that might prevent a successful upgrade.
- In order to do an upgrade, the SQL Server installer will need to have appropriate permissions to read and write to the server.

Question: When would you want to run two different versions of Reporting Services side by side?

Post Upgrade Tasks

Task	Description
Remove Old Files and Settings	<ul style="list-style-type: none">● Old log files● Obsolete RSWebApplication.config file● Virtual directory settings in IIS
Remove Applications from Previous Installation	<ul style="list-style-type: none">● SQL Server 2005 Report Designer● Management Studio

If an upgrade is done, the upgrade process will not delete old log files, the obsolete RSWebApplication.config file, or virtual directory settings in Internet Information Services (IIS). Upgrade will not remove SQL Server 2005 Report Designer, Management Studio, or other client tools. These will need to be removed manually. This is only relevant if an upgrade is done. If the user is deploying a side-by-side installation they will need to keep the files and applications for the prior installation on the computer.

Question: When would it be better to uninstall the previous version instead of upgrading?

Lab: Administering Reporting Services

- Exercise 1: Using Reporting Services Configuration Manager
- Exercise 2: Securing a Reporting Services Site
- Exercise 3: Securing Items

Logon information

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

Estimated time: 45 minutes

Exercise 1: Using Reporting Services Configuration Manager

Scenario

The Adventure Works team has asked you to publish a number of pre-written reports to a report server, and then to configure site security. You are also required to back up the site's encryption keys.

You will use the Reporting Services Configuration Manager to view Reporting Services settings and to back up the Reporting Services encryption keys.

The exercise's main tasks are:

1. Start the NY-SQL-01 virtual machine, log on as administrator, and set up the lab environment.
2. View Reporting Services settings using Configuration Manager.
3. Backup the Reporting Services encryption keys.

- ▶ **Task 1: Start the NY-SQL-01 virtual machine, log on as Administrator, and set up the lab environment**
 - Start NY-SQL-01 and log on as **Administrator** with the password of **Pa\$\$w0rd**.
 - Run the **E:\MOD08\Labfiles\Starter\runScripts.bat** file.
- ▶ **Task 2: View Reporting Services Settings**
 - Connect to the **MSSQLSERVER** instance on NY-SQL-01 in Reporting Services Configuration Manager.
 - View the **Report Server Status** page, and notice that the service is started.
- ▶ **Task 3: Back up the Reporting Services encryption keys**
 - Backup the **encryption keys** using Reporting Services Configuration Manager using **Pa\$\$w0rd** as the password.
 - Save the file to the **E:\Mod08\Labfiles\Starter** folder with the name **Keys**.

Results: After this exercise, you should have successfully viewed Reporting Services settings and backed up the Reporting Services encryption keys.

Exercise 2: Securing a Reporting Services Site

Scenario

You will deploy the Adventure Works sales reports to the report server. You will then secure the Reporting Services site using system-level security. You will create a new role that allows several tasks and link the role to a Windows group. Finally, you will test the role by using Report Manager as a user who is a member of the Windows group.

This exercise's main tasks are:

1. Open and deploy the Sales Reports solution.
2. Configure the AdventureWorks Data Source.
3. Open Report Manager as the NADirector user.
4. Create a new role.
5. Assign the new role to the AWSalesDirectors group.
6. View site settings as the NADirector user.

► Task 1: Open and deploy the Sales Reports solution

- Open **Sales Reports.sln** from E:\Mod08\Labfiles\Starter in SQL Server Business Intelligence Development Studio and deploy the solution.

► Task 2: Configure the AdventureWorks Data Source

- Browse to **http://NY-SQL-01/reports** in Internet Explorer® using **administrator** as the user name and **Pa\$\$w0rd** as the password.
- Click the **Sales Reports** folder, and then click the **AdventureWorks** data source.
- Configure the data source to use **Credentials stored securely in the report server** and **NY-SQL-01\Student** as the user name and **Pa\$\$w0rd** as the password.
- Set the data source to use the credentials as **Windows credentials**.

- ▶ **Task 3: Open Report Manager as the NADirector user**
 - Browse to <http://NY-SQL-01/reports> and log in as NY-SQL-01\NADirector using Pa\$\$w0rd as the password.
 - Note that the user has no permissions, and can therefore see no objects in Report Manager.
- ▶ **Task 4: Create a new role**
 - Open SQL Server Management Studio.
 - Open NY-SQL-01 as a **Reporting Services** server type using **Windows Authentication** as the authentication method.
 - Add a new System Role called **Security Manager**. Use **Can set item and site security** as the description.
 - Set the role to be able to manage report server security and manage roles only.
- ▶ **Task 5: Assign the new role to the AWSalesDirectors group**
 - Navigate to <http://NY-SQL-01/Reports> in Internet Explorer. Use **administrator** as the user name and **Pa\$\$w0rd** as the password.
 - Assign the **Security Manager** role to the NY-SQL-01\AWSalesDirectors group.
- ▶ **Task 6: View site settings as the NADirector user**
 - Navigate to <http://NY-SQL-01/Reports> in Internet Explorer. Use **NY-SQL-01\NADirector** as the user name and **Pa\$\$w0rd** as the password.
 - Verify that the Security section is the only section available in Site Settings.

Results: After this exercise, you should have successfully created the Security Manager role and applied that role to the AWSalesDirectors security group. You should then have successfully opened the Report Manager using the NADirector user (which is in the AWSalesDirectors group).

Exercise 3: Securing Items

Scenario

In this exercise, you will add item browsing permissions to the **AWSalesDirector** group by using the **Browser** role. Finally, you will test the role using Internet Explorer.

The exercise's main tasks are:

1. Assign the item permissions.
2. Browse reports as the NADirector user.

► Task 1: Assign the item permissions

- Navigate to **http://NY-SQL-01/Reports** in Internet Explorer. Use **administrator** as the user name and **Pa\$\$w0rd** as the password.
- Create a new folder called **Samples**.
- Assign the **NY-SQL-01\AWSalesDirectors** group the **Browser** role.
- Delete the **NY-SQL-01\AWSalesDirectors** group from the permissions of the **Samples** folder.

► **Task 2: Browse reports as the NADirector user**

- Browse to <http://NY-SQL-01/reports> and log in as **NY-SQL-01\NADirector** using **Pa\$\$w0rd** as the password.
- Note that the Samples folder link is not visible because the user has no permissions on the Samples folder.
- Click the **Sales Reports** folder link, and then click the **Company Sales** report link to view the report.
- Note that you cannot set any properties for the report because the user only has permission to browse.

Results: After this exercise, you should have successfully created a Samples folder using Report Manager. You should have given the AWSalesDirectors security group access to all the objects under the Home directory then removed the AWSalesDirectors security group from the permissions of the Samples folder.

Lab Shutdown

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

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips

Review Questions

1. How often do the Reporting Services encryption keys need to be backed up?
2. What are the two databases used by Reporting Services and what data do they contain?
3. What is the difference between “native” and “SharePoint Integration” authentication modes?
4. How many pre-defined Item roles are there and what does each one allow?
5. Which two previous versions of Reporting Services are supported for upgrade?

Common Issues and Troubleshooting Tips

Issue	Troubleshooting tip
Anticipating disk space	
Managing the report lifecycle.	
Managing multi-server environment complexity	
Managing permissions	

Module 9

Programming Reporting Services

Contents:

Lesson 1: Querying for Server Information Using a Web Service	9-3
Lesson 2: Automating Report Management	9-9
Lesson 3: Rendering Reports	9-13
Lesson 4: Creating Custom Code	9-22
Lab: Programming Reporting Services	9-26

Module Overview

- Querying for Server Information Using a Web Service
- Automating Report Management
- Rendering Reports
- Creating Custom Code

SQL Server Reporting Services offers several programming interfaces that you can leverage in your own applications. You can use the existing features and capabilities of Reporting Services to build custom reporting and management tools into Web sites and Windows applications, or you can extend the Reporting Services platform.

Extending the Reporting Services platform includes creating new components and resources that can be used for data access, report delivery and more. You can market these components and resources to companies that are using Reporting Services in their organization.

Lesson 1

Querying for Server Information Using a Web Service

- What Are Web Services?
- Web Service Functionality
- Using Web Services

Microsoft® SQL Server® Reporting Services provides access to the full functionality of the report server through the Report Server Web service. The Web service uses Simple Object Access Protocol (SOAP) over HTTP and acts as a communications interface between client programs and the report server. The Web service provides two endpoints - one for report execution and one for report management - with methods that expose the functionality of the report server and allow you to create custom tools for any part of the report life cycle.

What Are Web Services?

Allows communication between applications across the internet

- Follows platform-independent technology based on established standards
 - XML, SOAP, HTTP
- Allows interaction from any type of application
 - Microsoft Windows applications, Web applications, etc.

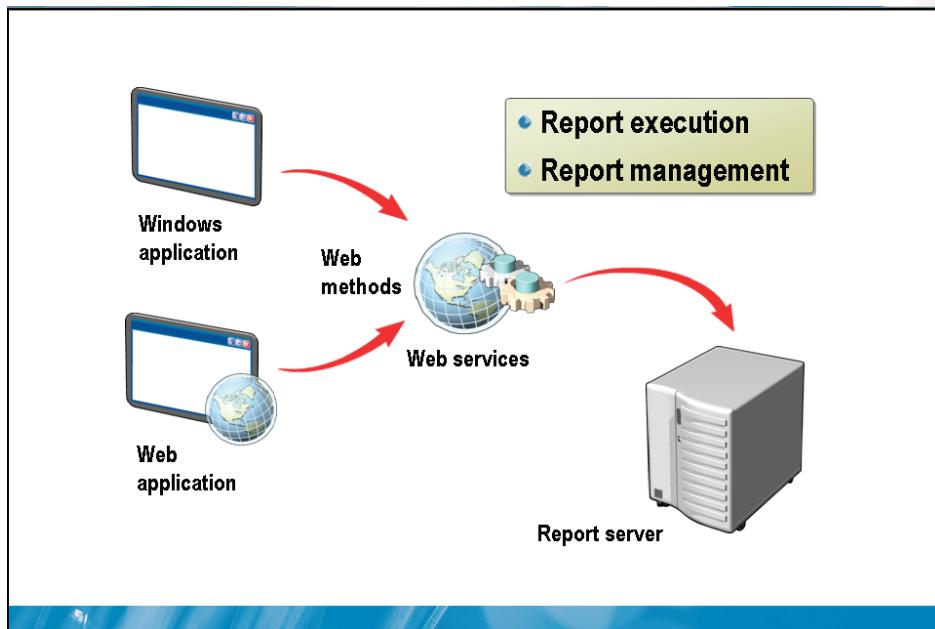
Key Points

Reporting Services uses Web services to enable you to view and manage reports. Before you look at the individual details of the Reporting Services Web services, you must understand the basics of the underlying technology.

- Web services enable communication between applications over open protocols by using standard data formats. They rely on the following standards to enable platform independence:
 - Extensible Markup Language (XML): Provides a standard data presentation
 - SOAP: Provides a standard format for packaging data
 - HTTP: Provides a standard protocol for sending and receiving data packets

- The above standards are set by the World Wide Web Consortium (<http://www.w3c.org>). Also known as the W3C, the World Wide Web Consortium is an independent organization composed of approximately 375 members.
- Any platform that can read and write XML and transport data over HTTP can use Web services. This flexibility provides the opportunity for integration with most applications platforms.
- Web services can be called from any type of application, including Microsoft® Windows® and Web applications.

Web Services Functionality



Key Points

Reporting Services provides a Web service that allows you to request and discover report server information.

- You can programmatically determine which reports are available for a particular user, whether those reports require parameters, and other pertinent information about the reports.
- In addition, you can programmatically change a report's attributes as part of an application to manage reports.
- The primary mechanism for getting information about Reporting Services items is by communicating with the Report Server Web service, which is accessible by using standard SOAP calls.

When a report server is configured for SharePoint® integrated mode and the Reporting Services Add-in has been installed, a set of proxy endpoints are installed on the Microsoft Office SharePoint Server. The endpoints that are installed are ReportService2006, ReportExecution2005, and ReportServiceAuthentication.

Using Web Services

Steps for using a Web service

- 1** Add a Web reference to the client application

```
http://<server>/reportserver/reportservice2005.asmx?wsdl  
(http://<Server Name>/<Site  
Name>/_vti_bin/ReportServer/ReportService2006.asmx?wsd  
l for SharePoint Integrated mode)
```
- 2** Specify security credentials for the proxy object

```
rs.Credentials =  
System.Net.CredentialCache.DefaultCredentials
```
- 3** Retrieve information as needed

Key Points

To retrieve server information about reports, you must follow some basic procedures to use the Web service. The following steps assume that you are using a Microsoft .NET Framework application as the client.

To create a Report Server Web service client using the .NET Framework, follow these basic steps:

- Create a proxy class for the Web service.
- Authenticate the Web service client with the report server.
- Call the method of the proxy class corresponding to the Web service operation that you want to invoke.

Question: What role do the Web Services Description Language (WSDL) files play in the Web service?

Demonstration: Creating a Web Services Proxy

In this demonstration, you will see how to:

- Create a Web Service proxy
- Create an instance of the proxy class

Question: How do you add an instance of the proxy class?

Lesson 2

Automating Report Management

- What Can Be Automated?
- Automation by Using Web Services
- Automation by Using Scripts

You can use automation to take care of the most common Reporting Services administrative tasks. Learning how to automate Reporting Services can provide a great deal of time savings over performing administration and maintenance tasks with the graphical management and configuration tools.

What Can Be Automated?

Automate any feature that Report Manager provides

- Add or delete reports and folders
- Administer security, schedules, etc.
- Examples of automation scenarios:
 - Apply identical report settings to multiple servers
 - Apply identical security settings to multiple reports

Key Points

Reporting Services provides automation capabilities by using the same Web service described in the preceding lesson for browsing information. Reporting Services supports the following configuration tasks.

- **Configure the Report Server Web service and Report Manager URLs.** You must write custom code that makes calls into the Report Server WMI provider.
- **Deploy existing content on another report server, including the folder hierarchy, role assignments, reports, subscriptions, schedules, data sources, and resources.** The best way to re-create an existing report server environment is to copy the report server database to a new report server instance.

Automation by Using Web Services

Use these method categories:

- **Create methods**
- **Delete methods**
- **Get methods**
- **List methods**
- **Set methods**

```
Dim report As Byte() = File.ReadAllBytes("Report.rdl")
Dim rs As New ReportingService2005()
rs.Credentials = System.Net.CredentialCache.DefaultCredentials
rs.CreateReport("My Report", "/Samples", True, report, Nothing)
```

```
ReportingService2005 rs = new ReportingService2005();
rs.Credentials =
System.Net.CredentialCache.DefaultCredentials;

DataSourceDefinition definition = new
DataSourceDefinition();
rs.SetDataSourceContents( "/SampleReports/AdventureWorks",
definition );
```

Key Points

To use the management Report Server Web service for automation, you must follow the same steps as for browsing server information described previously.

The Report Server Web services include several categories of methods that are based on component features.

- These methods are provided through two Web service endpoints (one for report management and one for report execution) which are exposed as members of the `ReportingService2005` and `ReportExecutionService` classes.
- These classes can be generated through a proxy class tool such as `wsdl.exe`, which is included with the Microsoft .NET Framework SDK.

Question: What happens if you attempt to upload a report that already exists through the `CreateReport` method with the `Overwrite` parameter set to `False`?

Automation by Using Scripts

- Use rs.exe as alternative to creating custom applications for management tasks

```
Public Sub Main()
    Dim items() As CatalogItem
    items = rs.ListChildren("/", True)

    For Each item As CatalogItem In items
        Console.WriteLine(item.Name)
    Next item
End Sub
```

```
rs.exe -i List.rss -s http://localhost/ReportServer
```

Key Points

Windows applications are very useful when you require complex user interaction to manage the report server. However, there will be times when you do not need this level of interactivity.

- Rather than write a complete executable application to carry out a management task, you can create a script file and then use the **rs.exe** utility to execute the script.
- The **rs.exe** utility requires a script file in order to manage a report server.
- The **rs.exe** utility expects an input file and a server URL as the minimum information. It will automatically use Windows authentication during the script execution.

Question: Can you think of a scenario in your organization where script automation of reports might be beneficial?

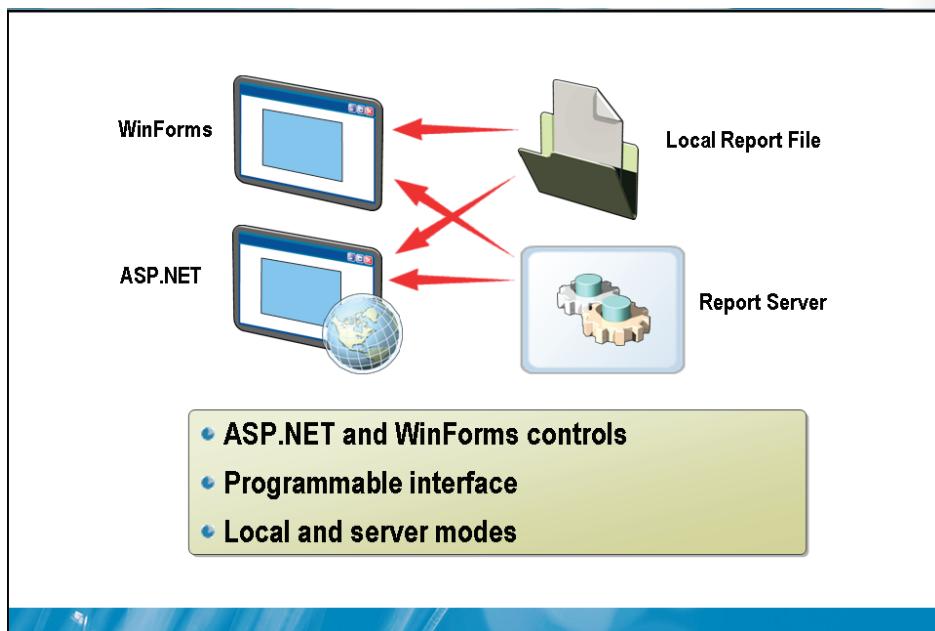
Lesson 3

Rendering Reports

- Using ReportViewer Controls
- Using URL Access
- Using Web Services
- Using Web Parts

Because report rendering is separate from the initial processing of the report data, the same report can be delivered in different formats—such as HTML, an Adobe PDF file, Microsoft Office Excel® spreadsheet—for different users. These different formats are supported by rendering extensions. Not only can reports be rendered in different formats, but through a variety of methods, allowing you great flexibility in rendering reports to meet different organizational needs and infrastructure requirements.

Using ReportViewer Controls



Key Points

Microsoft Visual Studio® 2008 contains two ReportViewer controls that you can use to display Reporting Services reports as part of a custom application.

- There are two versions of the ReportViewer control, one for Windows client applications and one for ASP.NET Web applications.
- Both versions of the ReportViewer control have a programmable interface, so you can customize, configure, and interact with the control through code at run time.
- Both ReportViewer controls support local and report server (remote) report processing.

- If you use local processing, your application provides the report dataset and report definition file and then processes the report on the client.
 - A Windows Forms application reads the report definition from a file and performs report processing on the client.
 - An ASP.NET application reads the report definition file from the Web site and performs report processing on the Web server.

Question: When might you choose local report processing?

Using URL Access

Internet and intranet sites can render reports in a browser

- Use a hyperlink from a Web page
`http://localhost/ReportServer?/Reports/Sales&rs:Command=Render`
- URL parameters control render behavior
`http://.../Sales&rs:Command=Render&rc:Toolbar=false&rc:Zoom=200`
- URL can include report parameters
- Use SSL to secure URL and parameters
- Web service detects appropriate HTML level for browser

Key Points

Displaying reports from a company Web site is one of the most common requirements for a report server. Any Web page can link directly to reports or folders by using the item's URL. This allows both Internet and intranet sites to render reports in a browser.

URL Access Through Direct Addressing

- To access a report server or report server database item using a URL simply provide the URL address from within a Web browser or application.
- You can also supply parameters to the URL that can affect the appearance of the report or resource that is being accessed.
- A URL can target a report server through the address bar of a Web browser, or a URL can be the source of an IFrame that is part of a larger Web application or portal.

URL Access Through a Form POST Method

- When a user requests data from a report server using URL access, the HTTP request uses the GET method.
- This is equivalent to a form submission where METHOD="GET". URL requests or form submissions that use METHOD="GET" are limited by the maximum number of characters that a server or Web browser can process.

Question: How would you view the Sales report with the toolbar, a zoom level of 125%?

Using Web Services

Use the ReportExecution2005 Web service

- Use the **LoadReport** method to specify the report path
- Use the **Render** method to:
 - Specify render format and other options
 - Returns byte array as result
 - Save byte array to file or display manually

Key Points

Using a URL is clearly the simplest way to render a report, but there are times when you want more control programmatically. You can render a report using the execution Report Server Web service in essentially the same way that you get report item information from the management Report Server Web service as previously described.

- The **ReportExecution2005** Web service provides the functionality to process and render reports. The URL should be in the format <http://server/reportserver/reportexecution2005.asmx?wsdl> and creates a **ReportExecutionService** proxy class.
- The **LoadReport** method needs to be called before you call the **Render** method. The main parameter required is the report path, but you can also supply a history id for a specific snapshot.

- The **Render** method has several parameters, including the rendering format. The function returns the rendered report as a byte array so that you can display the report in whatever way you require. For example, you can display an HTML rendered report in a literal Web control, or you can save the report output to a file.



Tip: For more information about these parameters, view the `ReportingExecutionService.Render` method definition in the SQL Server Books Online.

Question: In what order must the Web Services methods be called?

Using Web Parts

View reports and explore folders from SharePoint

- Use SharePoint 2.0 Web part for accessing reports from native mode report server
- Use SharePoint 3.0 Web part for accessing reports from SharePoint integrated report server
- Report Explorer Web Part browses report folders
- Report Viewer Web Part renders reports

Key Points

Reporting Services provides SharePoint Web parts so that you can view reports and explore report server contents from a SharePoint page.

- The **Report Explorer** Web Part enables you to navigate the report server and select a report to view.
- The purpose of the **Report Viewer** Web Part is to display a report. The default behavior is to work in connected mode.
 - This means that the **Viewer** Web Part displays whatever report whose link you click in the **Explorer** Web Part.
 - Alternatively, the Web Part can work in standalone mode and automatically display reports without user intervention.

Question: What is a custom web part and how can a developer create and deploy one?

Demonstration: Rendering a Report

In this demonstration, you will see how to:

- Utilize the Post Method for URL access
- Contrast to a comparable URL access string

Question: When would you want to use a Form code block over a URL access string?

Lesson 4

Creating Custom Code

- Extending Reporting Services
- What Are Custom Assemblies?

In Reporting Services, you can write custom code for report item values, styles, and formatting.

- For example, you can use custom code to format currencies based on locale, flag certain values with special formatting, or apply other business rules that are in practice for your company.
- One way to include this code in your reports is to create a custom code assembly using the Microsoft .NET Framework that you can reference from within your report definition files. The server calls the functions in your custom assemblies when a report is run.
- Custom assemblies can be used to retrieve specialized functions that you plan to use in your reports.

Extending Reporting Services

Extending Reporting Services

Create add-ins for Reporting Services

- Proprietary data sources
- Enhanced rendered output
(for example, 3-D view, link to diagrams)
- Extension types:
 - Data processing
 - Rendering
 - Delivery
 - Security

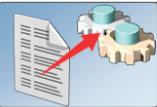
Key Points

Reporting Services provides you with most of the reporting features that you will require on a day-to-day basis. Occasionally, you might require functionality that the default installation does not provide. Fortunately, you can create additional functionality yourself, because Reporting Services is designed as an extensible architecture.

- The use of custom extensions allows you to create seamless add-ins for Reporting Services.
- For example, if you have a specialized application with a proprietary data source, you can create a custom data processing extension to retrieve data from your application.

- The major tasks that Reporting Service implements are extensions. Therefore, you can add additional customized extensions that integrate seamlessly with the native functionality of the report server.
- Currently, you can create new extensions based on the following extension types:
 - **Data Processing.** All data connections rely on a data processing extension. You can add your own data processing extension for your own data sources or, alternatively, you can add additional functionality to the existing Reporting Services data processing extensions.
 - **Rendering.** Each report render option is implemented as an individual rendering extension. You can create your own rendering extension to produce report output that matches your file types.
 - **Delivery.** Delivery extensions are responsible for delivering rendered reports. You can create your own delivery extension, such as an extension for a queuing application.
 - **Security.** Security extensions enable you to provide a custom security model that works with Reporting Services, such as a Forms authentication model.

What Are Custom Assemblies?



Create complex calculations and call from within a report



Access the full capabilities of the .NET platform



Logic is secure and reusable

Key Points

Reporting Services can take advantage of custom .NET Framework assemblies written by you or third-party developers. If you already have experience in programming in any .NET language, you will immediately be able to create custom .NET Framework assemblies to use in reports.

- Custom functions can be used to encapsulate complex or detailed expressions.
- Custom functions in an assembly can use the full capabilities of the .NET Framework.
- When you put custom functions in an assembly, you can be confident that the code is secure.

Lab: Programming Reporting Services

- Exercise 1: Using URL Access to Display a Report
- Exercise 2: Building a Reporting Services Web Service Client
- Exercise 3: Using the ReportViewer Control

Logon information

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

Estimated time: 60 minutes

Exercise 1: Using URL Access to Display a Report

Scenario

You need to integrate Reporting Services reports into an existing intranet Web site at Adventure Works. You also need to create a management utility that enables report administrators to view properties of reports and folders.

In this exercise, you will create a custom application that uses URL access to display a report.

The main tasks for this exercise are as follows:

1. Specify default values for a report URL.
2. Implement report invocation.
3. Configure toolbar and zoom settings.
4. Add report parameters.

► **Task 1: Specify default values for a report URL**

- Turn on 6236-NY-SQL-01 and logon as **Administrator**.
- Use Visual Studio 2008 to open the **ReportURL.sln** solution in the E:\Mod09\Labfiles\Starter\Exercise01 folder.
- Set the **Text** property of the following controls as follows:
 - **ReportServerTextBox**: http://ny-sql/reportserver
 - **FolderTextBox**: /AdventureWorks Sample Reports/
 - **ReportNameTextBox**: Company Sales

► **Task 2: Implement report invocation**

- Add the following constant declarations to the code page for the Default.aspx Web page:

```
Const RENDER_COMMAND As String = "&rs:Command=Render"
Const TOOLBAR As String = "&rc:Toolbar="
Const ZOOM As String = "&rc:Zoom="
```

- Add the following code to the **Button1_Click** event, under the **create report url** comment:

```
Dim url As String
url = ReportServerTextBox.Text + "?" + _
FolderTextBox.Text + ReportNameTextBox.Text + _
RENDER_COMMAND
```

- Under the **redirect to report page** comment, add the following code:

```
Response.Redirect(url, True)
```

- Test the Web page by running the project in Debug mode and clicking **Show Report**.

► **Task 3: Configure toolbar and zoom settings**

- Add items with the following **Text** property values to the combo box next to the **Show Toolbar** label:
 - True
 - False
- Add items with the following **Text** property values to the combo box next to the **Zoom** label:
 - 100
 - 200
- In the code file for the Web page, add the following code under the **configure toolbar and zoom** comment:

```
url += TOOLBAR + ToolbarDropDownList.Text + _  
ZOOM + ZoomDropDownList.Text
```

- Test the Web page by running the project in Debug mode and setting the **Show Toolbar** and **Zoom** options before clicking **Show Report**.

► **Task 4: Add report parameters**

- Add the following code underneath the **conditionally add parameter** comment:

```
If ParameterNameTextBox.Text.Length > 0 Then  
    url += "&" + ParameterNameTextBox.Text + "=" + _  
        ParameterValueTextBox.Text  
End If
```

- Test the Web page by running the project in Debug mode and viewing the **Employee Sales Summary** report with an **EmpID** parameter value of **285**.

Results: After this exercise, you should have created an ASP.net web site that allows you to enter report parameters and open the report using those parameters. You then modified the site to include toolbar options and finally should be able to view the Employee Sales Summary report for the employee with ID 285.

Exercise 2: Building a Report Services Web Service Client

Scenario

In this exercise, you will create a custom application that uses the Report Server Web service to view report information.

The main tasks for this exercise are as follows:

1. Add a Web reference to the Report Server Web service.
2. Add code to the form to retrieve server information.
3. Add code to retrieve individual item information.
4. Test the application.

► Task 1: Add a Web reference to the Report Server Web service

- Use Visual Studio 2008 to open the **ReportManager.sln** solution in the E:\Mod09\Labfiles\Starter\Exercise02 folder.
- Add a Web reference named **ReportService** for the Web service defined at the following URL:

<http://ny-sql-01/ReportServer/ReportService2005.asmx?wsdl>

► **Task 2: Add code to the form to retrieve server information**

- View **frmManage.vb** in code view.
- At the very top of the file, add an **Imports** statement to the **ReportManager.ReportService** namespace.
- After the existing **Inherits** statement, create a class-level variable called **rs** that references the **ReportingService2005** proxy class.
- Find the **frmManage_Load** event, and before the call to the **LoadInformation** method, add code to configure the **rs** proxy class to use the default credentials.
- Locate the **LoadInformation** method and modify the code as shown.

```
Private Sub LoadInformation()
    'load information into list view
    lstView.Items.Clear()
    Dim myCatalogItems As CatalogItem()

    myCatalogItems = rs.ListChildren("/", True)

    For Each cItem As CatalogItem In myCatalogItems
        Dim strValues(3) As String
        strValues(0) = cItem.Name
        strValues(1) = cItem.Path
        strValues(2) = cItem.Type.ToString()
        strValues(3) = cItem.CreatedBy

        lstView.Items.Add(New ListViewItem(strValues))
    Next
End Sub
```

- The code creates a list of all the items on the report server. It then loops through the list and displays the information in the list view control.

► **Task 3: Add code to retrieve individual item information**

- Find the **DisplayCurrentInfo** method.
- Modify the code as shown.

```
Private Sub DisplayCurrentInfo()
    Me.Cursor = Cursors.WaitCursor

    'display item information in messagebox
    Dim item As ListViewItem = 1stView.SelectedItems(0)
    Dim strName As String = item.SubItems(0).Text
    Dim strPath As String = item.SubItems(1).Text

    Dim properties As [Property]()
    properties = rs.GetProperties(strPath, Nothing)

    Dim sb As New System.Text.StringBuilder

    For Each prop As [Property] In properties
        sb.Append(prop.Name & ": " & prop.Value & vbCrLf)
    Next

    MessageBox.Show(sb.ToString(), _
                  "Properties of " & strName)
    Me.Cursor = Cursors.Default
End Sub
```

- The code retrieves information from the server about the item selected in the list view control. The information is then displayed in a message box.

► **Task 4: Test the application**

- Run the application in Debug mode.
- When the form loads (this could take a few seconds), it should display the list of all available items on the report server.
- Select one of the reports from the list view, and then click **Get Info**.
- Examine the properties displayed in the message box before closing it.
- Select one of the folders from the list view, and then click **Get Info**.
- Examine the properties displayed in the message box before closing it.

Results: After this exercise, you should have built a Reporting Services web service client using Visual Basic. You should know how to write a client application to consume web services from a report service and parse the information from using the service.

Exercise 3: Using the ReportViewer Control

Scenario

In this exercise, you will create a custom application that uses the Report Server Web service to view report information.

The main tasks for this exercise are as follows:

1. Add a ReportViewer control to a web page.
2. Configure the ReportViewer control.
3. Add report parameters.

► Task 1: Add a ReportViewer control to a web page

- Use Visual Studio 2008 to open the **ReportURL.sln** solution in the E:\Mod09\Labfiles\Starter\Exercise03 folder.
- Drag a **MicrosoftReportViewer** control from the toolbox and place it under the existing table on the Web page.
- Resize the control so that it is a suitable size to display a report in the window.

► Task 2: Configure the ReportViewer control

- Add the following **Imports** statement to the beginning of the code file for the Default.aspx Web page:

```
Imports Microsoft.Reporting.WebForms
```

- Add the following code under the **use remote processing** comment:

```
ReportViewer1.ProcessingMode = ProcessingMode.Remote  
Dim serverReport As ServerReport  
serverReport = ReportViewer1.ServerReport
```

- Add the following code under the **set the report server URL and report path** comment:

```
serverReport.ReportServerUrl = _
    New Uri(ReportServerTextBox.Text)
serverReport.ReportPath = FolderTextBox.Text + _
    ReportNameTextBox.Text
```

- Test the Web page by running the project in Debug mode and clicking **Show Report**.

► **Task 3: Add report parameters**

- Add the following code underneath the **conditionally add parameter** comment:

```
If ParameterNameTextBox.Text.Length > 0 Then
    Dim parameter As New ReportParameter()
    parameter.Name = ParameterNameTextBox.Text
    parameter.Values.Add(ParameterValueTextBox.Text)
    'set the report parameters for the report
    Dim parameters() As ReportParameter = {parameter}
    serverReport.SetParameters(parameters)
End If
```

- Test the Web page by running the project in Debug mode and viewing the **Employee Sales Summary** report with an **EmpID** parameter value of **288**.
- Turn off 6236-NY-SQL-01 and discard changes.

Results: After this exercise, you should have embedded the ReportViewer control in an ASP.net web page. You then should have modified the parameters of the control to allow you to view the Employee Sales Summary report.

Module Review and Takeaways

- Review Questions
- Common Issues and Troubleshooting Tips
- Best Practices
- Real-world Issues and Scenarios

Review Questions

1. What are the three primary ways to develop Reporting Services applications based on the Web service?
2. In what ways do Report Manager and a report server support SOAP?
3. Describe some of the ways the rs utility and the Web Service can be used to administer a report server.

Common Issues related to report access

Issue	Troubleshooting tip
I'm getting prompted to log in when I try to view a report I deployed	
I have to select the parameter from a dropdown before the report appears	

Real-world Issues and Scenarios

1. You need to pass data source credentials in a URL string securely. How would you do this?
2. You need to use a custom assembly from a third party to enable specific features in your reports. You also need to make sure that the assembly has execute permissions in your Reporting Services instance. How would you include the custom assembly in your deployment?

Best Practices related to URL Access

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

- If you need to pass hidden parameters that cannot be modified by users, or you have very long parameter strings, use the POST method for submitting URL's. The POST method supports an unlimited length URL and also supports hidden options.
- If your report contains sensitive data or parameters that should not be sent in clear text, use SSL to encrypt connections. Parameters passed through URL access are not encrypted by default.

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.

MCT USE ONLY. STUDENT USE PROHIBITED

Module 1: Introduction to Microsoft SQL Server Reporting Services

Lab: Using Reporting Services Tools

Exercise 1: Exploring Report Designer

► **Task 1: Open the Adventure Works Solution**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on as **Student** with the password of **Pa\$\$w0rd**.
3. Click **Start**, point to **All Programs**, point to **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
4. On the **File** menu, point to **Open**, and then click **Project/Solution**.
5. In the **Open Project** dialog box, browse to the E:\MOD01\Labfiles\Starter folder, and then double-click **AdventureWorks Sample Reports.sln**.

► **Task 2: Explore the Sales Order Detail Report**

1. In Solution Explorer, double-click **Sales Order Detail.rdl**.
2. Click the **Preview** tab. Notice that the report is a parameterized report that shows the detail of a particular sales order.

3. Click the **Design** tab. Notice that the report is made up of a few distinct areas:

- A header consisting of an Adventure Works image on the left, and a title on the right, together with an expression to display the report parameter value.
- A rectangle containing the Bill To, Ship To and order header details.
- A table region containing the order detail together with a footer containing totals.

Note: You can close the Output window, and Auto hide the Solution Explorer, Properties, Datasets, and Toolbox windows to make it easier to see the report.

► Task 3: Alter the Sales Order Detail Report

1. In the Toolbox, drag a **Textbox** control and drop it just above the text box containing the **Sales Order** title, aligned to the left edge.
2. Stretch the new textbox to the same width as the **Sales Order** title text box.
3. On the **Report Formatting** toolbar, click **Bold**, and then click **Align Right**.
4. Right-click the text box and then click **Expression**.
5. The **Expression** dialog box appears. In the **Set expression for: Value** field, type `=First(Fields!OrderDate.Value, "SalesOrder")` and then click **OK**.
6. In the **Properties** pane, in the **Number** section, in the **Format** field, type **d**.

► Task 4: Alter the Order Detail table region

1. On the report design surface, click the cell in the footer row of the **Discount** column of the table at the bottom of the report.
2. Right-click the text box and then click **Expression**.
3. The **Expression** dialog box appears. In the **Set expression for: Value** field, type `=Fields!UnitPriceDiscount.Value`, and then click **OK**.
4. In the **Properties** pane, type **p** in the **Format** property, and then press ENTER.
5. Click the **Preview** tab, and then review your changes.

► **Task 5: Change report parameter available values**

1. Click the **Design** tab.
2. In the **Report Data** pane, expand **Parameters**.
3. Right-click **SalesOrderNumber** and then click **Parameter Properties**.
4. In the **Report Parameter Properties** dialog box, click **Available Values**.
5. In the **Select from one of the following options** list, click **Specify values**.
6. For each of the following values, click **Add** and then specify the appropriate **Label** and **Value** fields:

Label	Value
SO50750	SO50750
SO50751	SO50751
SO50752	SO50752

7. Click **OK**.
8. Click the **Preview** tab.
9. In the **Sales Order Number** list, click **SO50751**, and then click **View Report**.
10. Close Business Intelligence Management Studio. Click **Yes** if you are prompted to save any items.

Results: After this exercise you should have a basic understanding of altering data and setting values for reporting queries.

Exercise 2: Exploring Report Manager

► Task 1: Start Report Manager

1. Click **Start**, point to **All Programs**, and right-click **Internet Explorer** and then click **Run as administrator**.
2. The **User Account Control** dialog box appears. Click **Allow**.
3. In the **Address** text box, type **http://NY-SQL-01/reports**, and then press **ENTER**.
4. The **Connect to NY-SQL-01** dialog box appears. Type the user name **Student** and the password **Pa\$\$w0rd** and then click **OK**.
5. The home page for Report Manager is displayed.

Note: If you get an **Information Bar** dialog box, click **Don't show this message again** and then click **Close**.

► Task 2: Examine the Product Catalog report

1. Click **AdventureWorks 2008 Sample Reports**.

Notice that the AdventureWorks 2008 Sample Reports folder contains the following reports:

- Company Sales 2008
- Employee Sales Summary 2008
- Product Catalog 2008
- Product Line Sales 2008
- Sales Order Detail 2008
- Territory Sales Drilldown 2008

2. Click the **Product Catalog 2008** link.

Notice that there is a document map on the left together with a number of catalog pictures on the right.

3. In the **Document Map**, if necessary, expand **Product Catalog**, expand **Bikes**, expand **Mountain Bikes**, and then click **Mountain-100**.
 4. In the **HTML viewer** toolbar, navigate through the report by clicking **Next Page**.
 5. In the **Current Page** box, type **11** and then press ENTER.
 6. In the **Find Text** box, type **saddle**, and then click **Find**. Then view the page that was found.
- **Task 3: View the Company Sales report**
1. On the upper left side of the screen, click the **AdventureWorks 2008 Sample Reports** link.
 2. Click the **Company Sales 2008** report link.
 3. Browse the report. Notice that you can expand the product categories and the years to drill-down and find more detailed information.
- **Task 4: Export the Company Sales report to Excel**
1. In the **Export** list, click **Excel**, and then click **Export**.
 2. In the **File Download** dialog box, click **Open**. The Sales Order report is now displayed as a Microsoft® Office Excel® file
 3. Close Excel. Click **No** if prompted to save any changes.

► **Task 5: Examine Company Sales report properties**

1. In Report Manager, click the **Properties** tab for the **Company Sales 2008** report.
2. The **General** tab is shown. Notice that the report name and description can be edited.
3. Click the **Data Sources** tab. Notice that the report uses the **AdventureWorks2008** shared data source located in the **/Samples/Data Sources** folder.
4. Close Microsoft Internet Explorer.
5. Turn off 6236A NY-SQL-01 virtual machine and discard changes.

Results: After this exercise, you should understand how to view reports through the Report Manager web site.

Module 2: Authoring Basic Reports

Lab: Authoring Basic Reports

Exercise 1: Creating a Basic Table Report

► **Task 1: Set up the lab environment**

1. In the Lab Launcher, next to **6236A-NY-SQL-01**, click **Launch**.
2. Log on as **Student** with the password of **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. Browse to the **E:\MOD02\Labfiles\Starter** folder, and then double-click **runScripts.bat**.
5. The script executes in a console window. Wait for the console window to close before proceeding to the next task.

► **Task 2: Create a Reporting Services project**

1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server® 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **New**, and then click **Project**.
3. In the **New Project** dialog box, in the **Project types** pane, ensure that the **Business Intelligence Projects** project type check box is selected.
4. In the **Templates** pane, click **Report Server Project**.
5. Change the Location to **E:\MOD02\Labfiles\Starter**.
6. In the **Name** text box, type **Product Reports**, and then click **OK**.

► **Task 3: Create a shared data source**

1. In Solution Explorer, right-click the **Shared Data Sources** folder, and then click **Add New Data Source**.

Tip: If Solution Explorer is not visible, on the **View** menu, click **Solution Explorer**.

2. In the **Shared Data Source Properties** dialog box, on the **General** tab, in the **Name** box, type **AdventureWorksDW2008**.
3. Click **Edit**, and then in the **Server name** box, type **localhost**.
4. Under **Log on to the server**, ensure that the **Use Windows Authentication** check box is selected.
5. In the **Select or ENTER a database** list, click **AdventureWorksDW2008**.
6. Click **Test Connection**, and then click **OK**.
7. Click **OK** to close the **Connection Properties** dialog box, and then click **OK** to close the **Shared Data Source Properties** dialog box.

► **Task 4: Create a report**

1. In Solution Explorer, right-click the **Reports** folder, point to **Add**, and then click **New Item**.
2. In the **Add New Item** dialog box, in the **Templates** pane, click **Report**. In the **Name** box, type **Product Profitability** and then click **Add**.

► **Task 5: Add a dataset**

1. In the **Report Data** pane, click **New**, and then click **Dataset**.
2. In the **Dataset Properties** dialog box, in the **Name** field, type **DataDetail**.

Caution: Ensure that there are no spaces in the name of your dataset.

3. In the **Data source** list, click **New**.

4. In the **Data Source Properties** dialog box, in the **Name** field, type **AdventureWorksDW2008**.
5. Click **Use shared data source reference**, and then, in the list, click **AdventureWorksDW2008**.
6. Click **OK**.
7. In the **Query** field, type:

```
SELECT * FROM vProductProfitability WHERE Year = 2003 AND  
MonthNumberOfYear = 1
```

8. Click **Query Designer**.

Note: You may see an error dialog box. This is not unusual; click OK to continue.

9. In the **Query Designer** window, click the **Run** button. A dataset containing the query results appears.
10. Click **OK** twice.
11. On the **File** menu, click **Save All**.

► **Task 6: Add a table data region**

1. Click the **Design** tab.
2. In the **Toolbox**, double-click **Table**. A table is added to the report. Notice that it has placeholder rows for the table header and table data.

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**.

► **Task 7: Add fields to the detail row**

1. In the **Report Data** pane, expand **DataDetail**, and then drag the **Product** field into the first cell in the **Data** row.
2. Adjust the **Product** column to be approximately **2** inches (5 cm) wide.
3. In the **Report Data** pane, drag the **SalesAmount** field into the second cell in the **Data** row.
4. Adjust the **Sales Amount** column to be approximately **2** inches (5 cm) wide.
5. In the **Report Data** pane, drag the **OrderQuantity** field into the third cell in the **Data** row.
6. Adjust the **Order Quantity** column to be approximately **1.5** inches (4 cm) wide.

► **Task 8: Add fields to the table footer**

1. In the **Design** pane, right-click the data row in the table, point to **Insert Row**, and then click **Outside Group - Below**.
2. Right-click the empty cell beneath **SalesAmount**, and then click **Expression**.
3. In the **Expression** dialog box, in the **Set expression for: Value** field, type:

```
=Sum(Fields!SalesAmount.Value)
```

4. Click **OK**.
 5. Right-click the empty cell beneath **OrderQuantity**, and then click **Expression**.
 6. In the **Expression** dialog box, in the **Set expression for: Value** field, type:
- ```
=Sum(Fields!OrderQuantity.Value)
```
7. Click **OK**.
  8. Right-click the empty cell beneath **Product**, and then click **Textbox Properties**.

9. In the **Text Box Properties** dialog box, in the **Value** field, type **Grand Total**, and then click **OK**.
10. On the **File** menu, click **Save All**.
11. Click the **Preview** tab. Notice that the report shows a header for each column and detail values from the source data set. A grand total is shown at the end of the report.

► **Task 9: Create a Product Category group**

1. Click the **Design** tab and then in the **Row Groups** pane, right-click **(Details)**, point to **Add Group**, and then click **Parent Group**.
2. In the **Tablix group** dialog box, set the **Group by** list to **Category**.
3. Click the **fx** button.
4. If the **Set expression for: GroupExpression** field is not set to **=Fields!Category.Value**, then type **=Fields!Category.Value**.
5. Click **OK**.
6. Select the **Add group header** and **Add group footer** checkboxes, and then click **OK**.
7. In the **Row Groups** pane, right-click **Group1**, then click **Group Properties**.
8. In the **Name** field, type **Category**, and then click **OK**.
9. In the **Design** pane, right-click the empty cell directly above **Sum(SalesAmount)**, and then click **Expression**.
10. In the **Set expression for: Value** field, type **=Sum(Fields!SalesAmount.Value)**, and then click **OK**.
11. Right-click the empty cell directly above **Grand Total**, and then click **Textbox Properties**.
12. In the **Text Box Properties** dialog box, in the **Value** field, type **Category Total**, and then click **OK**.
13. In the **Design** pane, right-click the empty cell directly above **Sum(OrderQuantity)**, and then click **Expression**.
14. In the **Set expression for: Value** field, type **=Sum(Fields!OrderQuantity.Value)**, and then click **OK**.

► **Task 10: Create a Product SubCategory group**

1. In the **Row Groups** pane, right-click **Category**, point to **Add Group**, and then click **Child Group**.
2. In the **Tablix group** dialog box, set the **Group by** list to **SubCategory**.
3. Click the **fx** button.
4. If the **Set expression for: GroupExpression** field is not set to **=Fields!SubCategory.Value**, then type **=Fields!SubCategory.Value**.
5. Click **OK**.
6. Select the **Show group header** and **Show group footer** checkboxes, and then click **OK**.
7. In the **Row Groups** pane, right-click **Group1**, and then click **Group Properties**.
8. In the **Name** field type **SubCategory**, and then click **OK**.
9. In the **Row Groups** pane, right-click **SubCategory**, and then click **Group Properties**.
10. In the **Group Properties** dialog box, click **Sorting**.
11. Verify that the **SubCategory** column is listed. If it is not, then click **Add**. In the **Sort by** list, click **SubCategory**.
12. Click **OK**.
13. In the empty cell directly above **Category Total**, type **SubCategory Total**.
14. In the **Design** pane, right-click the empty cell directly below **SalesAmount**, and then click **Expression**.
15. In the **Set expression for: Value** field, type  
**=Sum(Fields!SalesAmount.Value)**, and then click **OK**.
16. In the **Design** pane, right-click the empty cell directly below **OrderQuantity**, and then click **Expression**.
17. In the **Set expression for: Value** field, type  
**=Sum(Fields!OrderQuantity.Value)**, and then click **OK**.

► **Task 11: Apply basic formatting to the table**

1. Click the column handle (the gray box above the column) for the **Sales Amount** column.
2. In the **Properties** pane, in the **Number** section, in the **Format** property box, type **C0**, and then press ENTER. The cells format is set to currency with no decimal places.

**Tip:** By default, the Properties window is located on the right side of the screen. If the Properties window is not visible, on the **View** menu, click **Properties Window**.

3. Click the column handle for the **Order Quantity** column.
4. In the **Properties** pane, in the **Number** section, in the **Format** property text box, type **N0**, and then press ENTER. The cells format is set to numeric with no decimal places.
5. Click the **Detail** row icon (with three horizontal lines) to select the **Detail** row. In the **Properties** pane, in the **Fill** section, in the **BackgroundColor** list, click **More colors**, click **WhiteSmoke**, and then click **OK**.
6. Click the **Table Header** row icon, located on the topmost row. Then, at the **Table Footer** row icon, located on the bottom row, press CTRL+click. The header and footer rows are selected.
7. In the **Properties** pane, perform the following tasks:
  - Expand the **Font** section, and then, in the **FontSize** list, type **12pt**.
  - In the **FontWeight** list, click **Bold**.
  - In the **Fill** section, in the **Background Color** list, click **More colors**, click **Silver**, and then click **OK**.
8. Click the **Category Header** row icon, the second row from the top. At the **Category Footer** row icon, the second row from the bottom, press CTRL+click.

9. In the **Properties** pane, perform the following tasks:
  - Expand the **Font** section and then, in the **FontSize** list, type **12pt**.
  - In the **FontWeight** list, click **Bold**.
  - In the **Fill** section, in the **Background Color** list, click **More colors**, click **Gainsboro**, and then click **OK**.
10. Click the **SubCategory Header** row icon, the third row from the top. At the **SubCategory Footer** row icon, the third row from the bottom, press **CTRL+click**.
11. In the **Properties** pane, perform the following tasks:
  - Expand the **Font** section, and then, in the **FontSize** list, type **11pt**.
  - In the **FontWeight** list, click **Bold**.

► **Task 12: Indent row captions**

1. Click the cell that reads **[Product]**.
2. In the **Properties** pane, in the **Alignment** section, expand the **Padding** property group.
3. In the padding **Left** property text box, type **12pt**.
4. Click the cell that reads **SubCategory Total**, and then, in the **Properties** pane, in the padding **Left** property text box, type **12pt**.
5. On the **File** menu, click **Save All**.
6. Click the **Preview** tab. Verify that the report looks as you want it to. Correct any errors before proceeding.

**Results:** After this exercise, you should have created a report, added report data, and then added grouping and formatting to the report.

## Exercise 2: Formatting Report Pages

### ► Task 1: Open the Product Reports solution

**Note:** You can omit this task if you already have the Product Reports project open from Exercise 1.

1. In SQL Server Business Intelligence Development Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the **E:\MOD02\Labfiles\Starter\Product Reports** folder, and then double-click the **Product Reports.sln** solution.

### ► Task 2: Add a page break at the end of each category group

1. Ensure that the **Product Profitability** report is open in the Report Designer, and click the **Design** tab.
2. Click the table, right-click the **Category Footer** row icon, which is second from the bottom, point to **Row Group**, and then click **Properties**.
3. In the **Group Properties** dialog box, click **Page Breaks**.
4. In the **Change page break options** section, select the **Between each instance of a group** check box, and then click **OK**.

### ► Task 3: Add a report header to the first page of the report

1. On the **Design** tab, click the table so that you can see the row and column handles, and then click the handle in the top-left corner to select the table item.
2. Drag the table down so that the top of the table is approximately **1.5** inches (4 cm) from the top of the report.
3. In the Toolbox, click **Line**. Drag a horizontal line across the top of the report, aligned with the width of the table.
4. In the **Properties** pane, in the **Style** section, change the **LineWidth** property of the line (**line1**) to a value of **12pt**.

5. In the Toolbox window, click **Textbox**. On the report grid, click just under the line in the upper-left corner, and drag the text box to size it **2.5 inches** (6 cm) wide and **1.25 inches** (4 cm) high. Reposition and resize the text box as needed.

**Tip:** You can use the **Size**, **Width**, and **Height** properties in the **Properties** window to adjust the size of the text box. After an item is selected in Design view, you can use the arrow keys for large movements, or press CTRL and arrow keys for fine movements.

6. Click inside the text box, type **Adventure Works Product Profitability Report**.
7. With the text box selected, in the **Properties** pane, in the **Font** section, expand **Font**, and then, in the **FontSize** list, type **20pt**.
8. In the **FontWeight** list, click **Bold** to format the text box. Resize the text box as necessary so that you can see the full title on three lines of text.

► **Task 4: Add an image to the report**

1. In the **Toolbox** window, click **Image**. On the report grid, click to the right of the report title text box.
2. In the Image Properties dialog box, click Import, navigate to the **E:\MOD02\Labfiles\Starter folder**, click **logopart.jpg**, and then click **Open**.
3. Click **OK**.
4. Stretch and align the image as required to match the height of the **Report Title** text box.
5. On the **File** menu, click **Save All**.
6. Click the **Preview** tab. Notice that:
  - The report header and image appears on the first page only.
  - Each category appears on its own page.

► **Task 5: Add a repeating table header**

1. Click the **Design** tab.
2. Click the table, right-click the **Table Header** row icon, located on the first row, and then click **Tablix Properties**.
3. In the **Tablix Properties** dialog box, in the **Column Headers** section, select the **Repeat header columns on each page** checkbox, and then click **OK**.
4. Click the **Preview** tab, and notice that the product header appears in black on each report page.

► **Task 6: Add a page header to the report**

1. Click the **Design** tab.
2. Right-click the margin to the left of the report body, and then click **Add Page Header**. Notice that a new section appears at the top of the report.
3. In the Toolbox window, click **Textbox**.
4. Click inside the **Page Header**, and size the text box to **2 inches (5 cm)** wide and **0.25 inches (0.5 cm)** high.
5. Drag the text box to align it with the right edge of the page.
6. With the text box selected, in the **Properties** pane, in the **Alignment** section, in the **TextAlign** property list, click **Right**.
7. Click inside the text box, type **Product Profitability Report**.
8. In the **Properties** pane list, click **Page Header**.
9. In the **General** section, in the **PrintOnFirstPage** property list, set the value to **False**.
10. In the **General** section, in the **PrintOnLastPage** property list, set the value to **True**.
11. Click the **Preview** tab. Notice that the page header is printed on every page, except for the first page.

► **Task 7: Add a page footer to the report**

1. Click the **Design** tab.
2. Right-click the margin to the left of the report body, and then click **Add Page Footer**.
3. In the Toolbox, click **Textbox**.
4. Click inside the **Page Footer**, and size the text box to **1.5** inches (4 cm) wide and **0.25** inches (0.5 cm) high.
5. Click inside the text box, and then type **Company Confidential**.
6. With the text box selected, in the **Properties** pane, in the **Font** section, expand **Font**, and then in the **FontStyle** list, click **Italic**.
7. In the **Properties** pane, in the **Alignment** section, in the  **TextAlign** list, click **CENTER**.
8. In the **Properties** window list, click **Page Footer**.
9. In the **General** section, set the **PrintOnFirstPage** property to **True**, and set the **PrintOnLastPage** property to **True**.
10. On the **File** menu, click **Save All**.
11. Click the **Preview** tab, and verify that the report looks as you want it to. Notice that:
  - Each category is on a different page.
  - Table header and page footer are repeated on all pages of the report.
  - The page header appears on all pages except the first page.

**Results:** After this exercise, you have customized report formatting by adding page breaks, headers and footers, and an image.

## Exercise 3: Adding Calculated Values

### ► Task 1: Open the Product Reports solution

**Note:** You can omit this task if you already have the Product Reports project open from Exercise 2.

1. In SQL Server Business Intelligence Development Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the **E:\MOD02\Labfiles\Starter\Product Reports** folder, and then double-click the **Product Reports.sln** solution.

### ► Task 2: Create and add a calculated field for the profit margin

1. Ensure that the **Product Profitability** report is open in the Report Designer, and click the **Design** tab.
2. In the **Report Data** pane, right-click **DataDetail**, and then click **Dataset Properties**.
3. In the **Dataset Properties** dialog box, click **Fields**.
4. Click **Add**, and then click **Calculated Field**.
5. In the **Field Name** field, type **Margin**.
6. In the **Field Source** field, click the **Expression** button (**fx**).
7. In the **Expression** dialog box, in the lower-left pane, click **Fields (DataDetail)**; and then, in the lower-right pane, double-click **SalesAmount**. The field is added to the expression in the upper pane.
8. In the upper pane, place the cursor after **=Fields!SalesAmount.Value**, and then type a minus sign (-).
9. In the lower right pane, double-click **CostAmount**. You should now see the following expression in the upper pane:

```
=Fields!SalesAmount.Value - Fields!CostAmount.Value
```

10. Click **OK** twice. The **Margin** calculated field is added to the **DataDetail** dataset.
11. On the **File** menu, click **Save All**.

► **Task 3: Add the Margin calculated field to the report**

1. On the **Design** tab, right-click the column handle for the **Order Quantity** column, point to **Insert Column**, and then click **Right**. A new column appears.
2. In the **Datasets** window, drag the **Margin** field, into the **Detail** row in the new column.
3. In the **Design** pane, right-click the empty cell below the **[Margin]** field, and then click **Expression**.
4. In the **Set expression for: Value** field, type **=Sum(Fields!Margin.Value)**, and then click **OK**.
5. Select the expression in the **Margin** column in the **SubCategory Footer** row, and then press **CTRL+C** to copy it.
6. Click the empty cell in the **Margin** column in the **Category Footer** row. Press **CTRL+V** to paste the copied formula.

**Note:** Copying just the formula preserves the formatting in the destination cell.

7. Click the **Margin** cell in the **Category Total** footer, click it again to edit, select **[Margin]**, and then press **CTRL+C**.
8. Click the empty cell in the **Margin** column in the **Table Footer** row, which is the last row. Then, click the cell again to edit the empty formula. Press **CTRL+V** to paste the copied formula.
9. Click the column handle for the **Margin** column. In the **Properties** pane, in the **Number** section, in the **Format** field, type **C0**, and then press **ENTER**. The cell is formatted as currency with no decimal places.
10. On the **File** menu, click **Save All**.
11. Click the **Preview** tab, and verify that the report looks as you want it to. Fix any errors before continuing.

► **Task 4: Add a margin percentage expression to the detail row**

1. Click the **Design** tab.
2. Right-click the handle at the top of the **Margin** column, point to **Insert Column**, and then click **Right**. A new column appears.
3. In the last cell in the **Table Header** row, type **Margin %**.
4. Click the column handle for the **Margin %** column. In the **Properties** pane, in the **Number** section, in the **Format** field, type **P1**. The cells are formatted as a percentage with one decimal place.
5. Right-click the **Detail** cell of the **Margin %** column, and then click **Expression**.
6. In the **Expression** dialog box, type the following into the top box:

```
=Fields!Margin.Value / Fields!SalesAmount.Value
```

7. Click **OK**.
8. To view the new column, click the **Preview** tab.

► **Task 5: Add a margin percentage expression to the table footer**

1. Click the **Design** tab.
2. Click the **Sales Amount** cell in the **Table Footer** row, which is the last row. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **SalesAmount\_Total**.
3. Click the **Margin** cell in the **Table Footer** row, which is the last row. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **Margin\_Total**.
4. Right-click the **Margin %** cell in the **Table Footer** row, and then click **Expression**.
5. In the **Expression** dialog box, in the upper box, type :

```
=ReportItems!Margin_Total.Value /
ReportItems!SalesAmount_Total.Value
```

6. Click **OK**.
7. To view the added **Margin %** table footer, click the **Preview** tab, and then navigate to the end of the report.

► **Task 6: Add a margin percentage expression to the Category footer**

1. Click the **Design** tab.
2. In the **Category Footer** row, click the **Sales Amount** cell. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **SalesAmount\_Category**.
3. In the **Category Footer** row, click the **Margin** cell. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **Margin\_Category**.
4. Right-click the **Margin %** cell in the **Category Footer** row, and then click **Expression**.
5. In the **Expression** dialog box, in the **Expression** box, type:

```
=ReportItems!Margin_Category.Value /
ReportItems!SalesAmount_Category.Value
```
6. Click **OK**.
7. Click the **Preview** tab. Notice that a margin percent exists for each category.

► **Task 7: Add a margin percentage expression to the SubCategory footer**

1. Click the **Design** tab.
2. In the **SubCategory Footer** row, click the **Sales Amount** cell. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **SalesAmount\_SubCategory**.
3. In the **SubCategory Footer** row, click the **Margin** cell. In the **Properties** pane, in the **General** section, in the **Name** field, replace the existing name with **Margin\_SubCategory**.
4. Right-click the **Margin %** cell in the **SubCategory Footer** row, and then click **Expression**.
5. In the **Edit Expression** dialog box, in the upper box, type:

```
=ReportItems!Margin_SubCategory.Value /
ReportItems!SalesAmount_SubCategory.Value
```

6. Click **OK**.
7. Click the **Preview** tab. Notice that a value for margin percent now exists at subcategory level.

► **Task 8: Conditionally format rows with low margin percents**

1. Click the **Design** tab.
2. In the **Detail** row, click the **Margin %** cell.
3. In the **Properties** pane, in the **Font** section, in the **Color** property list, click **Expression**.
4. In the **Expression** dialog box, in the upper box, delete the default expression and type:

```
=IIF(Me.Value<0.15, "Red", "Black")
```

**Note:** The “Me” keyword refers to the report item for which the expression is being calculated. For a text box named **MarginPercent**, the expression Me.Value is equivalent to ReportItems!MarginPercent.Value.

5. Click **OK**.

► **Task 9: Copy conditional formatting to additional rows**

1. In the **Properties** pane, in the **Color** property list, double-click the expression, and then press CTRL+C to copy the expression.
2. In the **SubCategory Footer** row, click the **Margin %** cell.
3. In the **Properties** window, click the **Color** property, delete the existing value, and press CTRL+V to paste the expression.
4. Repeat steps 2 and 3 for the **Margin %** cell in the following rows:
  - Category Footer
  - Table Footer
5. Ensure that the **Margin %** cell in the **Table Footer** row is selected. In the **Properties** pane, in the **Color** property list, click **Expression**.

6. Modify the expression by replacing **Black** with **White**, and then click **OK**.
  7. On the **File** menu, click **Save All**.
  8. Click the **Preview** tab, and verify that the margin percent for AWC logo cap and the Caps SubCategory Total both appear in red.
- **Task 10: Sort products in descending order of Sales Amount**
1. Click the **Design** tab.
  2. In the **Row Groups** pane, right-click **(Details)**, point to **Add Group**, and then click **Parent Group**.
  3. In the **Tablix group** dialog box, in the **Name** field, type **DetailSort**.
  4. Click the **Expression** button.
  5. In the **Expression** dialog box, in the **Set expression for: GroupExpression** field, type **=Fields!Product.Value**, click **OK** twice.
  6. In the **RowGroups** pane, right-click **Group1**, and then click **Group Properties**.
  7. In the **Group Properties** dialog box, set the **Name** field to **DetailSort**, then click **OK**.
  8. In the **Design** pane, right-click **Group1**, and then click **Textbox Properties**.
  9. In the **Textbox Properties** dialog box, in the **Value** field, type **DetailSort** and then click **OK**.
  10. In the **Row Groups** pane, right-click **DetailSort**, and then click **Edit Group**.
  11. In the **Group Properties** dialog box, click **Sorting**.
  12. In the **Sorting** pane, click **Add**.
  13. In the **Sort by** list, click **SalesAmount**. In the **Order** column, click **Z to A**, and then click **OK**.
  14. On the **File** menu, click **Save All**.
  15. Click the **Preview** tab, and verify that the report looks as you want it to. Notice that the products are listed in descending order, based on sales amount.

► **Task 11: Show cumulative Sales Amount within SubCategory**

1. Click the **Design** tab.
2. Right-click the handle at the top of the **Sales Amount** column, point to **Insert Column**, and then click **Right**.
3. In the new cell in the **Table Header** row, type **Cumulative**.
4. In the **Detail** row, right-click the **Cumulative** cell, and then click **Expression**.
5. In the **Expression** dialog box, in the upper pane, type:  

```
=RunningValue(Fields!SalesAmount.Value, Sum, "SubCategory")
```
6. Click **OK**. Note that this expression calculates a running total of the sales amount, resetting each time the value of the **SubCategory** level changes.
7. In the **Properties** pane, in the **Number** section, in the **Format** field, type **C0**, and then press **ENTER**.
8. On the **File** menu, click **Save All**.
9. Click the **Preview** tab, and verify that the report looks as you want it to. Notice that the **Cumulative** value increases within each subcategory until it matches the total for the subcategory.

► **Task 12: Convert the Cumulative value to a percentage**

1. Click the **Design** tab.
2. In the **Detail** row, right-click the **Cumulative** cell, and then click **Expression**.
3. In the **Expression** dialog box, in the top pane, change the expression to:

```
=RunningValue(Fields!SalesAmount.Value, Sum, "SubCategory") /
ReportItems!SalesAmount_SubCategory.Value
```

4. Click **OK**. Note that this expression calculates the cumulative percentage of the subcategory for each product.
5. In the **Properties** pane, in the **Number** section, in the **Format** field, type **P1**, and then press **ENTER**.
6. Change the label in the **Table Header** row to **Cumulative %**.
7. On the **File** menu, click **Save All**.
8. Click the **Preview** tab, and verify that the report looks as you want it to. Notice that the cumulative percentages accumulate to 100% for each subcategory, and then restart with each new subcategory.
9. Turn off virtual machine and discard changes.

**Results:** After this exercise, you have added calculated fields, conditional formatting, and sorting to a report.

# Module 3: Enhancing Basic Reports

## Lab: Enhancing Basic Reports

### Exercise 1: Using Dynamic Visibility

► **Task 1: Setup the Lab environment**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on as **Student** with the password of **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. Browse to the E:\MOD03\Labfiles\Starter\Exercise01 folder, and then double-click **runScripts.bat**. 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: Open the AdventureWorks solution**

1. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, right-click **SQL Server Business Intelligence Development Studio** and then click **Run as administrator**.
2. The **User Account Control** dialog box appears. Click **Continue**.
3. The Microsoft Visual Studio window opens. On the **File** menu, point to **Open**, and then click **Project/Solution**.
4. In the **Open Project** dialog box, browse to the E:\MOD03\Labfiles\Starter\Exercise01 folder, click **AdventureWorks.sln**, and then click **Open**.
5. In **Solution Explorer**, under **Customer Sales**, expand **Reports**, and then double-click **Reseller Sales.rdl**.

► **Task 3: Preview the report**

1. Click the **Preview** tab to view the report. Notice that the report contains detail and summarized values for resellers, subcategories, and categories, including both header and footer summaries for subcategories and categories.

**Note:** You can close the Output window, and Auto Hide the Solution Explorer, Properties, Report Data, and Toolbox windows to make it easier to see the report.

2. Click the **Design** tab.

► **Task 4: Hide the Reseller detail and State footer rows**

1. Click the table data region, and then click the square to the left of the **detail** row (this row has [Reseller] in the left column and is the fifth row from the top). Clicking this should select the entire row.
2. In the **Properties** window, in the **Visibility** section, in the **Hidden** property list, click **True**.
3. In the table data region, click the square to the left of the **State Total** footer row.
4. In the **Properties** window, in the **Visibility** section, in the **Hidden** property list, click **True**.
5. On the **File** menu, click **Save All**.
6. Click the **Preview** tab. Notice that the detailed reseller rows and state footers are hidden.

► **Task 5: Toggle the visibility of the hidden rows**

1. Click the **Design** tab.
2. In the table data region, click the square to the left of the **detail** row (this row has [Reseller] in the left column and is the fifth row from the top).
3. In the **Properties** window, in the **Visibility** section, in the **ToggleItem** property list, click **State**.

4. In the table data region, click the square to the left of the **State Total** footer row.
5. In the Properties window, under the **Visibility** section, in the **ToggleItem** property list, click **State**.
6. On the **File** menu, click **Save All**.
7. Click the **Preview** tab. Expand one of the state totals by clicking the plus sign to the left of the state name.

► **Task 6: Show State totals when the detail is hidden**

1. Click the **Design** tab.
2. In the **State Total** footer row, right-click each of the following cells, click **Copy**, click the cell with the same name in the **State** header row (row 3) then press **CTRL+V** to paste it into the cell:
  - **Sales Amount**
  - **Cumulative %**
  - **Order Quantity**
  - **Margin**
3. Select each of the cells above individually, and then in the **Properties** window, make sure that the **Hidden** property is set to **False**, then expand the **Visibility** property, and then in the **ToggleItem** property list, click **State**.

**Important:** Make sure to change the **Hidden** property to **False** if needed.  
Toggling visibility will hide the data values from the fields.

4. On the **File** menu, click **Save All**.
5. Click the **Preview** tab.

6. Click the plus sign next to **Texas** to view the resellers for this state. Notice that the data next to Texas in the State header row is now hidden when you drill down.
7. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should have successfully previewed the report, hidden rows, toggled visibility, and preview the report after making changes.

## Exercise 2: Using Document Maps

### ► Task 1: Open the Document Map solution

1. In SQL Server Business Intelligence Development Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the E:\MOD03\Labfiles\Starter\Exercise02 folder, click **Document Map.sln**, and then click **Open**.
3. In **Solution Explorer**, under **Document Map**, expand **Reports**, and then double-click **Product Catalog.rdl**.

### ► Task 2: Preview the report

1. Click the **Preview** tab (ignore any warnings displayed when the report is built).
2. On the left side of the report, in the **Document Map**, if necessary expand **Product Catalog**, and then click **Clothing** to navigate to the clothing products.
3. In the **Document Map**, click **Bikes** to view the bike products.
4. In the **Preview** toolbar, navigate through the report by clicking **Next Page**.

► **Task 3: Add SubCategoryList to the document map**

1. Click the **Design** tab.
2. In the **Row Groups** pane, click **SubCategoryList Grouping** and then click **Group Properties**.

**Note:** If this is not visible, click **SubCategoryList** in the Properties window list. This is the drop-down list at the top of the Properties window.

3. If an error message appears, click **OK** and then repeat step 2.
4. In the **Group Properties** dialog box, verify that **[ProdSubCat]** is selected in the **Group on** list.
5. Click **Advanced**.
6. In the **Document map** list, click **[ProdSubCat]** and then click **OK**.

**Note:** By adding an entry in the Document map list, you are adding this report item to the document map.

► **Task 4: Add ModelList to the document map**

1. Click an empty area inside the **Design** pane.
2. In the **Properties** window list, click **ModelList**.
3. In the **Row Groups** pane, click **ModelList Grouping** and then click **Group Properties**.
4. If an error message appears, click **OK** and then repeat step 3.
5. In the **Group Properties** dialog box, verify that **[ProdModel]** is selected in the **Group on** list.
6. Click **Advanced**.
7. In the **Document map** list, click **[ProdModel]** and then click **OK**.

► **Task 5: Save and preview the report**

1. On the **File** menu, click **Save All**.
2. Click the **Preview** tab.
3. On the left side of the report, in the **Document Map**, under **Product Catalog**, expand **Clothing**, expand **Jerseys**, and then click **Short-Sleeve Classic Jersey**. Notice that subcategory and model groups are now contained in the document map and that the matching report item is displayed in the report. (You might need to scroll through the report to see the matching report item.)

**Note:** The document map can be hidden by clicking the **Show or Hide Document Map** button directly above it.

4. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should have successfully previewed the report, added a SubCategoryList to the document map, made changes to the document map, and previewed the report after making the changes.

### Exercise 3: Initiating Actions

► **Task 1: Set up the lab environment**

1. Click **Start**, and then click **Computer**.
2. Browse to the E:\MOD03\Labfiles\Starter\Exercise03 folder, and then double-click **runScripts.bat**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
3. Close the folder window.

► **Task 2: Open the Action solution**

1. In SQL Server Business Intelligence Development Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the E:\MOD03\Labfiles\Starter\Exercise03 folder, click **Action.sln**, and then click **Open**.

► **Task 3: Preview the report**

1. In the Solution Explorer, under **Reports**, double-click **Product Details.rdl**.
2. Click the **Preview** tab (ignore any warnings displayed when the report is built). Notice that the report contains information on individual products.
3. Click the **Design** tab.
4. On the **File** menu, click **Save All**.
5. In the Solution Explorer, under **Reports**, double-click **Order Details.rdl**.
6. Click the **Preview** tab (ignore any warnings displayed when the report is built). Notice that the report contains a series of orders for a single customer.
7. Click the **Design** tab.

► **Task 4: Add an action to jump to report**

1. In the **detail** row of the table (this row has [Product] in the left column and is the third row from the top), click the **Order** column.
2. In the **Properties** window, in the **Action** section, in the **Action** property, click the ellipsis.
3. In the **Text Box Properties** dialog box, click **Go to report**.
4. In the **Select a report from the list** list, click **Product Detail**.
5. In the **Use these parameters to run the report** section, click **Add**.
6. In the **Name** list, click **Product**.

7. In the **Value** list, click **Product**, and then click **OK**.
8. With the **Order** cell still selected, in the **Properties** window, in the **Font** section, expand **Font**.
9. In the **TextDecoration** list, click **Underline**. This will help to emphasize the report link.

► **Task 5: Save and preview the report**

1. On the **File** menu, click **Save All**.
2. Click the **Preview** tab. Click the product named **Road-650 Red, 60** in the report to jump to the Product Detail report. Notice that the report is filtered by product number.
3. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you have successfully previewed the report, added an action to the report, and saved and previewed the report after making changes.

## Exercise 4: Using a List Data Region

► **Task 1: Set up the lab environment**

1. Click **Start**, and then click **Computer**.
2. Browse to the E:\MOD03\Labfiles\Starter\Exercise04 folder, and then double-click **runScripts.bat**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
3. Close the folder window.

► **Task 2: Open the Data Regions solution**

1. In SQL Server Business Intelligence Development Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, navigate to the E:\MOD03\Labfiles\Starter\Exercise04 folder, click **Data Regions.sln**, and then click **Open**.

► **Task 3: Preview the report**

1. In Solution Explorer, under **Reports**, double-click **Table Product Profitability.rdl**.
2. Click the **Preview** tab. Notice that the report includes a table with a table header and footer, a category header and footer, and a subcategory header. The report contains no detail rows.
3. On the **File** menu, click **Close**. Click **No** when prompted to save changes.

► **Task 4: Copy the report**

1. In Solution Explorer, right-click **Table Product Profitability.rdl**, and then click **Copy**.
2. Press CTRL+V to paste a copy of the report.
3. Right-click the copy of the report, and then click **Rename**.
4. Type **List Product Profitability.rdl** and then press ENTER.
5. Double-click **List Product Profitability.rdl**.

► **Task 5: Expand the height of the report**

1. At the top of the **Properties** window, in the list, click **Body** to view the properties for the entire report body.
2. In the **Properties** window, in the **Position** section, expand the **Size** property group.
3. In the **Height** property, replace the existing value with **5 inches (12.5 cm)**.
4. In the Design window, click the table, and then click the table handle (at the top left of the table).

**Note:** Clicking the table handle will make it disappear and an icon (small square with 4 arrows) will appear. Use this icon when moving the table in the next step.

5. Drag the table to the bottom of the report.

► **Task 6: Add a Category list**

1. In the Toolbox, click **List**. If the Toolbox is not visible then on the **View** menu, click **Toolbox**.
2. Drag a list data region and position it **0.25** inches (0.5 cm) below the top edge of the report. Ensure that the list is the same width as the table and approximately **1.5** inches (4 cm) high.
3. Click the **List** data region, right-click the table handle that has the 3 lines on it, and then click **Tablix Properties**.

**Note:** This is the rectangle to the left of the list.

4. The **Tablix Properties** dialog box appears. In the **Dataset name** list, click **DataDetail** and then click **OK**.
5. Right-click the list data region, point to **Row Group** and then click **Group Properties**.
6. In the **Group Properties** dialog box, type **Category** in the **Name** field.
7. In the **Group expressions** section, click **Add**.
8. In the **Group on** list, click **Category**.
9. Click **Sorting**.
10. In the **Change sorting options** section, click **Add**.
11. In the **Sort by** list, click **Category**, and then click **OK**.

**Note:** With a Table data region, you can right-click a row handle to add a new grouping level. With a List data region, there is no option for adding a new grouping level.

► **Task 7: Add fields to the Category list**

1. In the **Report Data** pane, under **DataDetail**, drag the **Category** field and then drop it inside the bottom left corner of the list.
2. Drag the right side of the **First(Category)** text box until the text box is approximately **1.875** inches (5 cm) wide. Resize and reposition as needed to move the text box to the upper left corner of the list, but not in the header section of the list.
3. In the **Report Data** pane, drag the **SalesAmount** field and drop it to the right of the **First(Category)** text box inside the list. Resize the **Sum(SalesAmount)** text box so that it is approximately **1.3** inches (4 cm) wide.
4. With the **Sum(SalesAmount)** text box still selected, in the **Properties** window, in the **Number** section, in the **Format** property, type **C0** to format the numbers as currency with 0 decimal places.
5. In the Report Data window, drag an **OrderQuantity** field and drop it to the right of the **Sum(SalesAmount)** field inside the list. Resize the **Sum(OrderQuantity)** text box so that it is approximately **1.3** inches (4 cm) wide.
6. With the **Sum(OrderQuantity)** text box still selected, in the **Properties** window, in the **Number** section, in the **Format** property, type **N0** to format the information as numbers with 0 decimal places.

► **Task 8: Add header and footer rows**

1. SHIFT+ click the **First(Category)**, **Sum(SalesAmount)**, and **Sum(OrderQuantity)** text boxes to select all of them. Right-click the selected fields and then click **Copy**.
2. Press CTRL+V to paste the fields on to the design surface. Drag the three fields separately to be positioned just above the list in the same order as they appear in the list.
3. Press CTRL+V to paste the fields on to the design surface. Drag the three fields separately to be positioned just below the list in the same order as they appear in the list.

4. SHIFT+click the six new text boxes outside the list, and then in the Properties window, perform the following tasks:
  - In the **Font** section, in the **FontSize** list, type **12pt**.
  - In the **Font** section, click **Color**, and then click **White**.
  - In the **Fill** section, in the **BackgroundColor** list, click **Black**.
5. In the **First(Category)** text box above the list, replace the text with **Category**.
6. In the **First(Category)** text box below the list, replace the text with **Grand Total**.
7. In the **Sum(SalesAmount)** text box above the list, replace the text with **Sales Amount**.
8. In the **Sum(OrderQuantity)** text box above the list, replace the text with **Order Quantity**.
9. On the **File** menu, click **Save All**.
10. Click the **Preview** tab. Notice that the list contains all product categories together with a grand total.

**Note:** If you receive warnings about overlap, move the appropriate text boxes to remove the overlap and preview the report again.

#### ► Task 9: Add SubCategory list

1. Click the **Design** tab.
2. In the Toolbox window, click **List**.
3. Click and drag a List below the **Category** list, leaving a **0.25** inches (0.5 cm) gap between the lists. Resize the list to be the same width as the **Category** list and approximately **0.25** inches (0.5 cm) high.
4. Click the new list data region, right-click the table handle that has the 3 lines on it, and then click **Tablix Properties**.
5. The **Tablix Properties** dialog box appears. In the **Dataset name** list, click **DataDetail** and then click **OK**.
6. Right-click the new list data region, point to **Row Group** and then click **Group Properties**.

7. In the **Group Properties** dialog box, type **SubCategory** in the **Name** field.
8. In the **Group expressions** section, click **Add**.
9. In the **Group on** list, click **SubCategory**.
10. Click **Sorting**.
11. In the **Change sorting options** section, click **Add**.
12. In the **Sort by** list, click **SubCategory** and then click **OK**.

► **Task 10: Add fields to the SubCategory list**

1. In the **Report Data** pane, drag the **SubCategory** field and drop it into the **SubCategory** list. Drag the text box **0.25** inches (0.5 cm) from the left edge inside the **SubCategory** list, and resize the text box to approximately **1.75** inches (4.5 cm) wide.
2. In the **Category** list, SHIFT+click the **Sum(SalesAmount)** and the **Sum(OrderQuantity)** details text boxes (both are found in the row with a white background), right-click the selected text boxes, and then click **Copy**.
3. Press **CTRL+V** to paste the text boxes on to the design surface.
4. Drag the two text boxes individually to the **SubCategory** list, to the right of the **First(SubCategory)** field.
5. Ensure that the **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes are horizontally aligned.
6. On the **File** menu, click **Save All**.
7. Click the **Preview** tab. Notice that there are two lists: the first has a header and footer and contains categories; the second contains a list of subcategories.

► **Task 11: Combine the Category and SubCategory lists**

1. Click the **Design** tab.
2. Drag the subcategory list, and drop it inside the category list, just below the row containing the **First(Category)**, **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes. Ensure that the **Sum(SalesAmount)** and **Sum(OrderQuantity)** columns in the subcategory list are aligned with the **Sum(SalesAmount)** and **Sum(OrderQuantity)** columns in the categories list.

The report should now contain:

- A row of text boxes containing the text **Category**, **Sales Amount**, and **Order Quantity**.
  - A list data region that contains:
    - A row of text boxes with the values **First(Category)**, **Sum(SalesAmount)** and **Sum(OrderQuantity)**.
    - A second list data region containing row of text boxes, which in turn contain the values **First(SubCategory)**, **Sum(SalesAmount)**, and **Sum(OrderQuantity)**.
  - A row of text boxes under the category list containing the values **Grand Total**, **Sum(SalesAmount)**, and **Sum(OrderQuantity)**.
  - The table from the original **Table Product Profitability.rdl** report.
3. Select the **First(Category)**, **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes in the **Category** list by SHIFT+clicking them. Right-click the selected text boxes and then click **Copy**.
  4. Press CTRL+V to paste the text boxes onto the design surface.
  5. Drag the text boxes individually to the space between the **SubCategory** row and above the **Grand Total** row. Ensure that you drop the copied text boxes below the subcategory list, but still in the category list. (You might need to resize the category list and move the total text boxes under the list if there is not enough space.)

6. Remove the **Sum(SalesAmount)** and **Sum(OrderQuantity)** text boxes in the first row of the **Category** list (the totals are now displayed in the row under the subcategory list instead).
7. In the second **Category** list (under the **SubCategory** list), click **First(Category)**.
8. Replace the existing text with **Category Total**.

The report should now contain:

- A row of text boxes containing the text **Category**, **Sales Amount**, and **Order Quantity**.
- A list data region that contains:
  - A text box with the value **First(Category)**.
  - A second list data region containing row of text boxes, which in turn contain the values **First(SubCategory)**, **Sum(SalesAmount)**, and **Sum(OrderQuantity)**.
  - A row of text boxes with the values **Category Total**, **Sum(SalesAmount)** and **Sum(OrderQuantity)**.
- A row of text boxes under the category list containing the values **Grand Total**, **Sum(SalesAmount)**, and **Sum(OrderQuantity)**.
- The table from the original **Table Product Profitability.rdl** report.

► **Task 12: Delete the table version of the report**

1. Click the table at the bottom of the report, and then press DELETE.
2. On the **File** menu, click **Save All**.
3. Click the **Preview** tab to preview the completed report.
4. Close the solution.
5. Turn off 6236A-NY-SQL-01 virtual machine and discard changes.

**Results:** After this exercise, you should have successfully added category lists, header and footer rows, subcategory lists, and combined the category and subcategory lists.

## Module 4: Manipulating Data Sets

# Lab: Lab Manipulating Data Sets

### Exercise 1: Using Parameters to Restrict Query Results

► **Task 1: Set up the lab environment**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on to **NY-SQL-01** as **Administrator** using the password **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**
4. Browse to the E:\MOD04\Labfiles\Starter\Exercise01 folder, and then double-click **runScripts.bat**. 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: Open the Data Manipulation solution**

1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server® 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to the E:\MOD04\Labfiles\Starter\Exercise01 folder, click **Data Manipulation.sln**, and then click **Open**.
4. In Solution Explorer, expand the **Data Manipulation** project, expand **Reports**, and then double-click **Product Profitability.rdl**.
5. Click the **Preview** tab. Notice that the report is hard-coded to display data for January 2003. The user cannot change the date used in the report.

**Note:** You can close the Output window, and Auto-hide the Solution Explorer, Properties, Datasets, and Toolbox window to make it easier to see the report.

► **Task 3: Modify the data set query**

1. Change back to **Design** view by clicking on the **Design** tab.
2. On the **View** menu, click **Report Data**.
3. Expand **AdventureWorksDW2008**, click **DataDetail**, and then on the **Report Data** toolbar, click the **Edit** button. The **Dataset Properties** dialog box opens.
4. In the left pane, click **Query**.
5. In the **Query** field, replace the numeric constants for **Year** and **Month** with the parameters **@Year** and **@Month**. You should now have the following SQL statement:

```
SELECT *
FROM vProductProfitability
WHERE (Year = @Year) AND (MonthNumberOfYear = @Month)
```

6. Click **OK**.

► **Task 4: Edit the Year and Month parameters**

1. On the **Report Data** pane, expand **Parameters**. Notice that the **Month** and **Year** report parameters have been created automatically.
2. Perform the following steps to edit the **Year** parameter:
  - a. Double-click **Year**.
  - b. Click **Default Values**, and select the **Specify values** option. A **Default Values** text box appears.
  - c. Click **Add**, and then type **2003** in the new **Value** line that appears.
  - d. Click **OK**.

**Tip:** If you want the default value of the **Year** parameter to be dynamically based upon the current year, use the expression **=Year(Now)**.

3. Perform the following steps to edit the **Month** parameter:
  - a. Double-click **Month**.
  - b. Click **Available Values**, and select the **Specify values** option. A **Default Values** text box appears.
  - c. For each of the values below, click **Add** and type the label and value into the new line:

| Label | Value |
|-------|-------|
| Jan   | 1     |
| Feb   | 2     |
| Mar   | 3     |
| Apr   | 4     |
| May   | 5     |
| Jun   | 6     |

- d. Click **OK** to close the Report Parameter Properties window.

**Tip:** If you want the default value of the **Month** parameter to be dynamically based upon the current month, use the expression **=Month(Now)**.

4. Close the **Report Data** pane.

► **Task 5: Save and preview the report**

1. On the **File** menu, click **Save All**.
2. Click the **Preview** tab. Notice that a **Year** text box and a **Month** drop-down list appear at the top of the report.
3. In the **Month** list, click **Jun**, and then click **View Report**.
4. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should have created parameters and set values for selection.

## Exercise 2: Using Parameters to Filter Report Data

► **Task 1: Open the Data Manipulation solution**

1. If Business Intelligence Development Studio is not already open, click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **E:\MOD04\Labfiles\Starter\Exercise02** folder, click **Data Manipulation.sln**, and then click **Open**.
4. In Solution Explorer, expand the **Data Manipulation** project, expand **Reports**, and then double-click **Product Profitability.rdl**.

► **Task 2: Add a data set for the Category parameter**

1. On the **View** menu, click **Report Data**.
2. Right-click **AdventureWorksDW2008**, click **Add Dataset**.
3. In the **Dataset** dialog box, in the **Query** pane, in the **Name** text box, type **Category**.
4. In the **Data source** list, verify that the data source is **AdventureWorksDW2008**.

5. In the **Query** text box, type the following SQL statement:

```
SELECT DISTINCT EnglishProductCategoryName AS Category
FROM DimProductCategory
```

6. Click **Query Designer**.
7. Click the **Run** button to execute the query. Correct any errors before continuing.
8. Click **OK**.
9. On the **Fields** screen, verify that a line exists with the value of **Category** in both text boxes exists. If it does not, click **ADD**, and type **Category** into both of the text boxes.
10. Click **OK**.

► **Task 3: Create the Category parameter**

1. In the **Report Data** pane, right-click **Parameters** and then click **Add Parameter**.
2. Perform the following steps to add the **Category** parameter:
  - a. On the **General** screen, in the **Name** box, type **Category**.
  - b. In the **Prompt** box, type **Category**.
  - c. Verify that the **Data type** is **Text**.
  - d. Clear the **Allow blank value** check box if it is checked.
  - e. Click **Available Values**.
  - f. On the **Available Values** page, click **Get values from a query**.
  - g. In the **Dataset** drop-down list, click **Category**.
  - h. In the **Value field** drop-down list, click **Category**.
  - i. In the **Label field** drop-down list, click **Category**.
  - j. Click **Default Values**.
  - k. In the **Default values** page, click **Specify values**, and then click **Add**.
  - l. In the new **Value** text box, type **Components**.
  - m. To close the **Report Parameter Properties** dialog box, click **OK**.

► **Task 4: Add a filter to the table data region**

1. On the **Design** tab, click inside the table data region.
2. Right-click on the border that appears.
3. Click **Tablix Properties**.
4. In the **Tablix Properties** dialog box, click **Filters**.
5. Click **Add**, and then in the **Expression** list, click **Category**.
6. In the **Operator** list, verify that the equal sign (=) appears.
7. Next to the **Value** text box, click the **fx** button.
8. In the **Expression** dialog box, in the **Category** pane, click **Parameters**, and then in the **Values** pane, double-click **Category** to add it to the **Expression** pane.
9. Verify that the resulting expression is as follows:

```
=Parameters!Category.Value
```

10. Click **OK** to close the **Expression** dialog box, and then click **OK** again to close the **Tablix Properties** dialog box.

► **Task 5: Remove the Category group page break**

1. Click inside the table data region.
2. Right-click the row that has **Category Total** in the far left column, point to **Row Group**, and then click **Group Properties**.
3. Click **Page Breaks**.
4. On the **Page Breaks** screen, clear the **Between each instance of a group** checkbox, and then click **OK**.
5. On the **File** menu, click **Save All**.
6. Click the **Preview** tab. Notice the **Category** parameter defaults to the **Components** category.
7. In the **Category** list, select **Bikes**, and then click **View Report**. Verify that the report shows sales for the **Bikes** category.

► **Task 6: Add the Category to the report header**

1. Click the **Design** tab.
2. From the Toolbox, drag a text box into the space above the bicycle graphic. Resize the text box to match the text box above.

**Note:** the Toolbox can be accessed by pointing at the **Toolbox** icon along the left side of the screen, or on the **View** menu click on **Toolbox**.

3. Right-click the new text box and then click **Expression**.
4. In the **Category** pane, click **Parameters**, and then in the **Values** pane, double-click **Category** to add it to the **Expression** pane.
5. Verify that the resulting expression is as follows:

```
=Parameters!Category.Value
```

6. To close the **Expression** dialog box click **OK**.
7. On the **File** menu, click **Save All**.
8. Click the **Preview** tab. Notice that the current value of the **Category** parameter is displayed above the graphic.
9. In the **Month** list, click **Mar**. In the **Category** list, click **Bikes**, and then click **View Report**. Verify that the report is updated to display sales of bikes in March.
10. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should have created parameters and used them to filter report data.

## Exercise 3: Creating Dynamic Parameter Lists

### ► Task 1: Open the Dynamic Parameters solution

1. If Business Intelligence Development Studio is not already open, click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **E:\MOD04\Labfiles\Starter\Exercise03** folder, click **Dynamic Parameters.sln**, and then click **Open**.
4. In Solution Explorer, expand the **Dynamic Parameters** project, expand **Reports**, and then double-click **Product Details.rdl**.
5. Click the **Preview** tab. Notice that the **SubCategory** parameter is used to restrict data in the report. The default value for the **SubCategory** parameter is **Mountain Bikes**.

► **Task 2: Add a data set for the Category parameter**

1. Click the **Design** tab.
2. Click the **Report Data** tab. If the Report Data pane is not available, on the **View** menu, click **Report Data**.
3. At the top of the **Report Data** list, click **New**, and then click **Dataset**.
4. In the **Dataset Properties** dialog box, on the **Query** screen, in the **Name** field, type **Category**.
5. In the **Data source** list, verify that the data source is **DataSet1**.
6. In the **Query** field, type the following SQL statement:

```
SELECT ProductCategoryID, Name
FROM Production.ProductCategory
```

7. Click **Query Designer**.
8. Click the **Run** button to execute the query. Correct any errors before continuing.
9. Click **OK**.
10. To close the **Dataset Properties** dialog box click **OK**.

► **Task 3: Modify the SubCategory data set query**

1. In the **DataSet1** list, double-click **SubCategory**.
2. In the **Dataset Properties** dialog box, in the **Query** field, add the following WHERE statement to the end of the statement, verify the statement, and then click **OK**:

```
WHERE (ProductCategoryID = @ProductCategory)
```

The following SQL statement should be displayed in the SQL field:

```
SELECT ProductSubcategoryID, ProductCategoryID, Name
FROM Production.ProductSubcategory
WHERE (ProductCategoryID = @ProductCategory)
```

► **Task 4: Modify the ProductDetail data set query**

1. In the **Dataset1** list, double-click **ProductDetail**.
2. In the **Dataset Properties** dialog box, in the **Query** field, edit the SELECT statement to add the code in bold text:

```
SELECT PS.Name AS ProdSubCat, PM.Name AS ProdModel, PC.Name AS
ProdCat, PD.Description, PP.LargePhoto, P.Name AS ProdName,
P.ProductNumber,
 P.Color, P.Size, P.Weight, P.StandardCost,
P.Style, P.Class, P.ListPrice, PS.ProductCategoryID
FROM Production.Product AS P INNER JOIN
 Production.ProductSubcategory AS PS INNER
JOIN
 Production.ProductCategory AS PC ON
PS.ProductCategoryID = PC.ProductCategoryID ON
 P.ProductSubcategoryID =
PS.ProductSubcategoryID INNER JOIN
 Production.ProductProductPhoto AS PPP ON
P.ProductID = PPP.ProductID INNER JOIN
 Production.ProductPhoto AS PP ON
PPP.ProductPhotoID = PP.ProductPhotoID LEFT OUTER JOIN
 Production.ProductDescription AS PD INNER
JOIN
 Production.ProductModel AS PM INNER JOIN
Production.ProductModelProductDescriptionCulture AS PMPDCL ON
PM.ProductModelID = PMPDCL.ProductModelID ON
 PD.ProductDescriptionID =
PMPDCL.ProductDescriptionID ON P.ProductModelID = PM.ProductModelID
WHERE (PMPDCL.CultureID = @Language) AND
(PS.ProductSubcategoryID = @SubCategory) AND (PS.ProductCategoryID
= @ProductCategory)
```

3. Click **Query Designer**.
4. Click **Run**.

5. In the **Define Query Values for query parameters** dialog box, enter the following parameter values and then click **OK**:

| Parameter Name   | Parameter Value |
|------------------|-----------------|
| @Language        | English         |
| @SubCategory     |                 |
| @ProductCategory | 3               |

6. Verify that there are no errors. Fix any errors. Note that the result set may be empty.
7. To close the **Query Designer** dialog click **OK**.
8. To close the **Dataset Properties** dialog click **OK**.
9. On the **File** menu, click **Save All**.

► **Task 5: Edit the ProductCategory parameter**

1. On the **Report Data pane**, expand **Parameters**. Notice that the **ProductCategory** parameter has been created automatically.
2. Click **ProductCategory**.
3. If **SubCategory** is listed above **ProductCategory**, click the **up-arrow** button to move **ProductCategory** above **SubCategory**.
4. Double-click **ProductCategory**.
5. In the **Report Parameter Properties** dialog box, in the **Data type** drop down, click **Integer**.
6. In the **Available Values** pane, click **Get values from a query**.
7. In the **Dataset** list, click **Category**.
8. In the **Value field** list, click **ProductCategoryID**.
9. In the **Label field** list, click **Name**.
10. In the **Default values** screen, click **Specify Values** and then click **Add**.
11. In the **Value** field, type **1**.
12. Click **OK** to close the **Report Parameter Properties** dialog box.

► **Task 6: Set the title of the report**

1. On the **Design** tab, right-click the text box that contains the text **[&ReportName]**, and then click **Expression**.
2. In the **Edit Expression** dialog box, edit the expression as follows, and then click **OK**:

```
=Globals!ReportName & " for " & Parameters!ProductCategory.Label1
```
3. On the **File** menu, click **Save All**.
4. Click the **Preview** tab, and then in the **ProductCategory** list, click **Clothing**. Notice that the **SubCategory** parameter is a dynamic parameter list dependent on the **ProductCategory** parameter.
5. In the **SubCategory** list, click **Shorts**, and then click **View Report**. Verify that the report displays product details for shorts.
6. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should be able to use dynamic parameters to display a different list of available values in the second parameter based on the selected value in the first parameter.

## Exercise 4: Using Parameters with a Stored Procedure

► **Task 1: Set up the lab environment**

1. Click **Start**, and then click **Computer**.
2. Browse to the E:\MOD04\Labfiles\Starter\Exercise04 folder, and then double-click **runScripts.bat**. The script executes in a console window. Wait for the console window to close before proceeding to the next step
3. Close the folder window.

► **Task 2: Open the Stored Procedure solution**

1. If Business Intelligence Development Studio is not already open, click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **E:\MOD04\Labfiles\Starter\Exercise04** folder, click **Stored Procedure.sln**, and then click **Open**.
4. In Solution Explorer, expand the **Stored Procedure** project, expand **Reports**, and then double-click **Actual Vs Quota.rdl**.

► **Task 3: Create a report data source**

1. If the **Report Data** window is not open, on the **View** menu, click **Report Data**.
2. At the top of the pane, click **New**, and then click **Data Source**.
3. In the **Name:** field, type **AdventureWorks2008**.
4. Select **Use shared data source reference** and then in the dropdown list select **AdventureWorksDW2008**.
5. Click **OK**.

► **Task 4: Create a stored procedure data set**

1. At the top of the **Report Data** pane, click **New**, and then click **Dataset**.
2. In the **Dataset Properties** dialog box, on the **Query** pane, in the **Name** field, type **Detail**.
3. In the **Data source** list, ensure that **AdventureWorks2008** is displayed.
4. Select **StoredProcedure** under **Query type**.
5. Click **Query Designer**.
6. If **Query Designer** opens with a **Command** type drop down at the top, click **Edit As Text**.
7. In the **Stored procedure** list, click **sp\_ActualVsQuota**, and then click the **Run** button to open the **Define Query Parameters** dialog box.

8. In the **CalendarYear** Parameter Value text box, type **2003**.
9. In the **Group** Parameter Value text box, type **Europe** and then click **OK**.
10. Click **OK** to close the **Query Designer**, and then click **OK** again to close the **Dataset Properties** dialog.
11. On the **File** menu, click **Save All**.

► **Task 5: Edit report parameters**

1. On the **Report Data** pane.
2. Expand **Parameters**.
3. Double-click **CalendarYear** and perform the following steps to edit the **CalendarYear** parameter:
  - a. In the **Available values** section, ensure that **None** is selected.
  - b. In the **Default values** section, select the **Specify values** option.
  - c. Click **Add** and type **2003** into the text box.
  - d. Click **OK**.
4. Double-click **Group** and perform the following steps to edit the parameter:
  - a. In the **Available values** section, ensure that **None** is selected.
  - b. In the **Default values** section, select the **Specify values** option.
  - c. Click **Add**, and in the text box, type **North America** into the text box.
  - d. Click **OK**.
5. Close the **Report Data** pane.

► **Task 6: Save and preview the report**

1. On the **File** menu, click **Save All**.
2. If the **Design** tab is not active then click the **Design** tab.
3. Click the tablix with **Sales Representative** at top left.
4. Right-click the border that appears around the tablix and click **Tablix Properties**.
5. Under **Dataset name** select **Detail**, and click **OK**.
6. Click the **Preview** tab.
7. In the **Group** parameter text box, type **Europe**, and then click **View Report**. Verify that the report updates to show sales performance in Europe.
8. On the **File** menu, click **Close Project**.

**Results:** After this exercise, you should have learned how to create data sets and assign default values to report parameters.

**Exercise 5 (If Time Permits): Displaying All Items in a Parameter List**

► **Task 1: Set up the lab environment**

1. Click **Start**, and then click **Computer**.
2. Browse to the E:\MOD04\Labfiles\Starter\Exercise05 folder, and then double-click **runScripts.bat**. The script executes in a console window. Wait for the console window to close before proceeding to the next step
3. Close the folder window.

► **Task 2: Open the All Items solution**

1. If Business Intelligence Development Studio is not already open, click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, navigate to the **E:\MOD04\Labfiles\Starter\Exercise05** folder, click **All Items.sln**, and then click **Open**.
4. In Solution Explorer, expand the **All Items** project, expand **Reports**, and then double-click **All Items Product Profitability.rdl**.
5. Click the **Preview** tab. Notice that the report contains the **Category** parameter, which has a default value of **Accessories** and a list from which you can select.
6. In the **Category** parameter list, click **Bikes**, and then click **View Report**. Notice that there is no way to display all categories in the same report.
7. Click the **Design** tab.

► **Task 3: Add all categories to the Category data set**

1. On the **View** menu, click the **Report Data**.
2. In the **Dataset** list, double-click **Category**.
3. In the **Query** text box, add the following code above the **ORDER BY** clause:

```
UNION SELECT -1 AS Expr1, 'All Categories' AS Expr2
```

4. Click the background of the **Report Designer** pane.

You should have the following SQL statement in the **Query** text box:

```
SELECT DISTINCT ProductCategoryKey AS CategoryKey,
EnglishProductCategoryName AS Category
FROM DimProductCategory
UNION
SELECT -1 AS Expr1, 'All Categories' AS Expr2
ORDER BY CategoryKey
```

5. Click **Query Designer**, and then click the **Run** button to see the results of the query. Notice that the All Categories entry has been added to the query with a key value of -1, and then click **OK**.
6. Click **OK**.
7. On the **File** menu, click **Save All**.

► **Task 4: Update the DataDetail data set query**

1. In the **Dataset** list, double-click **DataDetail**.
2. In the **Query** field, add the following code to the **WHERE** clause:

```
OR @Category = - 1
```

The following SQL statement should be displayed in the **SQL** pane:

```
SELECT Product, SubCategory, CategoryKey, Category, CostAmount,
SalesAmount, OrderQuantity, Month, MonthNumberOfYear, Year
FROM vProductProfitability
WHERE (Year = 2003) AND (MonthNumberOfYear IN (1, 2, 3))
AND (CategoryKey = @Category OR @Category = - 1)
```

**Note:** Make sure that you are consistent with the capitalization of @Category.

3. Click **OK**.
4. On the **File** menu, click **Save All**.
5. Click the **Preview** tab.
6. In the **Category** parameter list, click **All Categories**, click **View Report**, and then verify that all categories are displayed in the report.
7. On the **File** menu, click **Close Project**.
8. Turn off NY-SQL-01 and discard changes.

**Results:** After this exercise, you should have successfully added the "All Items" option to the dynamic parameter to allow the display of all possible values for selection.

## Module 5: Using Report Models

# Lab: Working with Report Models

### Exercise 1: Creating a Report Model

► **Task 1: Start the Virtual Machine**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on to **NY-SQL-01** as **Administrator** with the password of **Pa\$\$w0rd**.

► **Task 2: Create a Report Model project**

1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server® 2008**, and click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **New**, and then click **Project**.
3. In the **Templates** list, click **Report Model Project**.
4. In the **Name** box, type **AdventureWorks Report Model**.
5. In the **Location** box, type **E:\MOD05\Labfiles\Solution\Exercise01**, and then click **OK**.

► **Task 3: Create a Data Source**

1. In Solution Explorer, right-click **Data Sources**, and then click **Add 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, verify that **Create a data source based on an existing or new connection** is selected, and then click **New**.
4. In the **Connection Manager** dialog box, in the **Server name** text box, type **NY-SQL-01**.
5. Ensure that **Use Windows Authentication** is selected.
6. In the **Select or enter a database name** list box, click **AdventureWorks2008**.
7. To verify that the connection works, click **Test Connection**, and then click **OK**.
8. Click **OK** to close the **Connection Manager** dialog box, and then click **Next**.
9. On the **Completing the Wizard** page, ensure that the **Data source name** is set to **Adventure Works2008**, and then click **Finish**.

**Note:** To edit the properties of an existing data source, double-click the data source in the Data Sources folder to display the data source properties in Data Source Designer.

► **Task 4: Create a Data Source View**

1. In Solution Explorer, right-click **Data Source Views**, and then click **Add 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, in the **Relational data sources** pane, verify that the **Adventure Works2008** data source is selected, and then click **Next**.
4. On the **Select Tables and Views** page, in the **Available objects** list, expand the **Name** column header so that you can see the table names.
5. In the **Available objects** list, click the following objects while holding the **CTRL** key:
  - **Person(Person)**
  - **Customer(Sales)**
  - **SalesOrderHeader(Sales)**

**Note:** If the object names are partially obscured, stretch the Data Source View Wizard window.

6. Click **>** to add the objects to the **Included objects** list, and then click **Next**.
7. On the **Completing the Wizard** page, in the **Name** text box, type **Customer Sales**, and then click **Finish**.

► **Task 5: Create a Report Model**

1. In Solution Explorer, right-click **Report Models**, and then click **Add New Report Model**.
2. On the **Welcome to the Report Model Wizard** page, click **Next**.
3. On the **Select Data Source View** page, in the **Available data source views** list, verify that **Customer Sales.dsv** is selected, and then click **Next**.
4. On the **Select report model generation rules** page, view the default rules, and then click **Next**.
5. On the **Collect Model Statistics** page, ensure that **Update model statistics before generating** is selected, and then click **Next**.

6. On the **Completing the Wizard** page, verify that the **Name** of the report model is **Customer Sales**, and then click **Run**.
7. When the report model has been generated, click **Finish**.
8. If you are prompted to reload the **Customer Sales.dsv** file, click **Yes to All**.
9. In the **Model** tree, click **Sales Order Header**.
10. In the **Properties** window, in the **Name** text box, type **Sales Order**, and then in the **CollectionName** text box, type **Sales Orders**.
11. In the **Attributes** pane, right-click **#Sales Order Headers**, click **Rename**, type **#Sales Orders**, and then press **ENTER**.
12. On the **File** menu, click **Save All**.

► **Task 6: Publish the Report Model to a Report Server**

1. In Solution Explorer, right-click the **AdventureWorks Report Model** project, and then click **Properties**.
2. In the **AdventureWorks Report Model Property Pages** dialog box, change the value of the **TargetServerURL** property to **http://ny-sql-01/ReportServer**, and then click **OK**.
3. In **Solution Explorer**, right-click the **AdventureWorks Report Model** project, click **Deploy**, and then wait for deployment to complete.
4. Close **SQL Server Business Intelligence Development Studio**.
5. On the **Start** menu, point to **All Programs**, and then click **Internet Explorer**.
6. In the **Address** text box, type **http://ny-sql-01/reports**, and then press **ENTER**.
7. If asked for credentials login as **Administrator** with password of **Pa\$\$w0rd**.
8. If you are asked to change intranet settings, click **Enable Intranet Settings**.
9. Click the **Models** folder. Notice that **Customer Sales** model has been deployed to the Reports Server.

**Results:** After this exercise, you should have created a report model, customized it for easier use, and published the model to the report server.

## Exercise 2: Using Report Builder to Create and Execute a Report

### ► Task 1: Creating a Tabular Report

1. If Report Manager is not already open, then using Windows® Internet Explorer®, browse to <http://ny-sql-01/reports>.
2. Click the **Report Builder** button.
3. In the **Application Run – Security Warning** dialog box, click **Run**. There may be a short delay while the **Report Builder** application is downloaded and run on your computer.
4. In the **Getting Started** pane in **Report Builder**, in the list of data sources ensure that **Customer Sales** is selected.
5. In the **Report Layout** section, ensure **Table (columnar)** is selected, and then click **OK**.

### ► Task 2: Add Fields and Entities to the Report

1. In the **Explorer** pane, click the **Person** entity in the **Entities** list.
2. In the **Fields** list, drag the **Last Name** attribute to the **Drag and drop column fields** area in the design area.
3. Click the **Last Name** text box, and then replace the text with **Name**.
4. Right-click the cell directly below the **Name** text box, and then click **Edit Formula**.
5. In the **Formula for each Person** text box, append the text **& ", " &** after **Last Name** and then drag the **First Name** field from the **Fields** list to the end of the expression so that the expression matches the following text, then, click **OK**:

Last Name & ", " & First Name

6. Resize the **Name** text box to a width of approximately **1.5** inches (4 cm).
7. In the **Entities** list, click **Customers**, and then click **Sales Orders**.
8. In the **Fields** list, drag the **#Sales Orders** attribute to the table region, to the right of the **Name** column.

9. In the **Fields** list, drag the **Total Total Due** attribute to the table region, to the right of the **#Sales Orders** column.
10. Click the **Total Total Due** text box, and then replace the text with **Total Value**.
11. Click the two table cells beneath the **Total Value** text box.
12. While holding the SHIFT key, click both cells the **Total Value** data region.
13. Right-click the **Total Value** data region and then click **Format**.
14. In the **Format** dialog, in the **Format** list, click **\$1,234.56**, and then click **OK**.
15. Stretch the **Total Value** text box to a width of approximately **1.25** inches (3.5 cm).

► **Task 3: Define the Sort Order**

1. Click the **Sort and Group** button.
2. In the **Sort by** list, click **Total Value**, and then choose the **Descending** option.
3. In the upper **Then by** list, click **Name**, and then click **OK**.

► **Task 4: Define the Report Filter**

1. Click the **Filter** button.
2. In the **Entities** list, click **Customers**, and then click **Sales Orders**.
3. From the **Fields** list, drag **Order Date** to the filter area.
4. Click **equals**, and then click **From...To**.
5. In the **from** box, type **10/1/2003**.
6. In the **to** box, type **12/31/2003**.
7. Click **OK**.

► **Task 5: Format the Report**

1. In the design area, click **Click to add title**, and then click the text box again to edit the title.
2. Type **Customer Sales**, and then press ENTER.
3. On the **Format** menu, click **Fill**.
4. In the **Color** palette, click **Cornflower**.
5. On the **Font** tab, in the **Color** palette, click **White**, and then click **OK**.
6. On the **Insert** menu, click **Filter Description**.
7. Drag the text box containing the filter description and place it directly beneath the **Customer Sales** text box.
8. Stretch the filter text box to the same width as the **Customer Sales** text box.

► **Task 6: Run the Report**

1. Click the **Run Report** button. Notice that the report is ordered by **Total value**, with totals shown at the end of the report. Notice that the filtering expression is shown at the beginning and end of the report.
2. Click **Design Report**.

► **Task 7: Save the Report to the Report Server**

1. On the **File** menu, click **Save**.
2. In the **Save As Report** dialog box, ensure that **http://ny-sql-01/ReportServer** is selected in the **Look in** list.
3. In the **Name** box, type **Customer Sales**, and then click **Save**.
4. Close **Report Builder**.

► **Task 8: View the Report**

1. Using Internet Explorer, browse to <http://ny-sql-01/reports>. If this page is already open, click the **Home** link at the top.
2. On the **Home** page, click **Customer Sales**. The report displays. If the report is not listed on the **Home** page then click **Refresh(F5)**.
3. Notice that the report is first sorted by the **Total Value** field, then by **Name**.
4. Click the **up-arrow** next to **Name** to change the sort order.
5. Click **Abercrombie, Kim**. Notice that **Person** entity from the report model is shown.
6. Close Internet Explorer.
7. Turn off NY-SQL-01 and discard changes.

**Results:** After this exercise, you should have created a report using a report model, formatted the report, published it to the report server, and viewed the report.

# Module 6: Publishing and Executing Reports

## Lab: Publishing and Executing Reports

### Exercise 1: Publishing Reports

► **Task 1: Set up the lab environment**

1. In the Lab Launcher, next to **6236A-NY-SQL-01**, click **Launch**.
2. Log on to **NY-SQL-01** as **Administrator** with the password of **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. Browse to the **E:\MOD06\Labfiles\Starter\Exercise01** folder, and then double-click **runScripts.bat**. 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: Open the solution**

1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server® 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to the **E:\MOD06\Labfiles\Starter\Exercise01\AdventureWorks** folder, click **AdventureWorks.sln**, and then click **Open**.
4. In Solution Explorer, browse the three projects that make up the Adventure Works solution and note the following:
  - The **Product Sales** project has three reports that use two shared data sources.
  - The **Territory Sales** project has three reports that use report-specific data sources.
  - The **Reseller Sales** project has three reports that use report-specific data sources.

**Tip:** If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**.

► **Task 3: Deploy the solution**

1. In Solution Explorer, right-click the **Product Sales** project, and then click **Properties**. In the **Product Sales Property Pages** dialog box, under **Deployment**, in the **TargetServerURL** text box, type **http://ny-sql-01/ReportServer**, and then click **OK**.
2. In Solution Explorer, right-click the **AdventureWorks** solution, and then click **Deploy Solution**.
3. In the **Output** window, confirm that all three projects were successfully deployed without warnings.
4. Keep Business Intelligence Development Studio open.

► **Task 4: Browse the published reports**

1. Open Internet Explorer and browse to Report Manager at **http://ny-sql-01/reports**. If credentials are requested use **Administrator** for the user name and a password of **Pa\$\$w0rd**.
2. If a warning about intranet settings appears at the top of the screen, click the warning and then click **Enable Intranet Settings**.
3. Click **Yes** on the warning dialog that appears.
4. Notice that there is one new folder for each project in the Adventure Works solution.
5. Click the **Product Sales** folder link. Notice that the **Product Sales** folder contains two data sources and three reports.
6. Click the **Product Profitability** report link and browse the report.
7. Close Internet Explorer.

► **Task 5: Update the Product Profitability report**

1. Return to Business Intelligence Development Studio. In Solution Explorer, double-click the **Product Profitability.rdl** report to open the report in the Report Designer.
2. In the **Toolbox**, click **Textbox**. In the **Page Footer**, create a text box next to the existing **Company Confidential** text box. Resize the new text box to the right edge of the page.

**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 inside the new text box and type the following:
  - **=Globals!ExecutionTime**
4. Select the text box, and on the **Report Formatting** toolbar, click the **Italic** button, and then click the **Align Right** button.
5. Click the **Preview** tab, and ensure that the page footer contains the current date and time.

6. On the **File** menu, click **Save All**.
7. In Solution Explorer, right-click the **Product Profitability** report, and then click **Deploy**.
8. In the **Output** window, scroll up to view the **Product Profitability** report deployment information. Even though the build and deploy succeeded, notice the warnings that the data source could not be deployed because it already exists and the **OverwriteDataSources** property is not specified.
9. To modify the **OverwriteDataSources** property, in Solution Explorer, right-click the **Product Sales** project, and then click **Properties**.
10. In the **Product Sales Property Pages** dialog box, under **Deployment**, change the **OverwriteDataSources** property to **True**, and then click **OK**.
11. In Solution Explorer, right-click the **Product Profitability** report, and then click **Deploy**. Notice that the deployment warnings are not displayed this time.
12. Close Business Intelligence Development Studio.

► **Task 6: Browse the updated Product Profitability report**

1. Open Internet Explorer and browse to Report Manager at <http://ny-sql-01/reports>. If Report Manager is already open then click the **Home** link at the top of the page.
2. Click the **Product Sales** folder link, and then click the **Product Profitability** report link and browse the report.
3. Keep Report Manager open.

**Results:** After this exercise, you should have published a report solution, made updates to a report in the solution and republished the report.

## Exercise 2: Executing a Report on Demand

► **Task 1: Configure the Order Details report for on-demand execution**

1. In Report Manager, click the **Home** link.
2. Click the **Reseller Sales** folder link, and then click **Order Details**. Notice that the **Order Details** report requires a start date, an end date, and a reseller to execute.
3. Click the **Properties** tab, and then click **Data Sources** on the left side of the screen. Notice that the custom **AdventureWorks2008** data source is using Microsoft® Windows® integrated security. This means that the current Windows account is used to connect to the Microsoft SQL Server® database.
4. Under **Connect using**, select the **Credentials supplied by the user running the report** option.
5. Select the **Use as Windows credentials when connecting to the data source** check box.
6. Click **Apply**.
7. On the left side of the screen, click the **Execution** link. Notice that the **Order Details** report is set up to always run with the most recent data. Execution on demand is the default execution setting.

► **Task 2: View the Order Details report**

1. Click the **View** tab.
2. In the **LogIn Name** text box, type **NY-SQL-01\Student**.
3. In the **Password** text box, type **Pa\$\$w0rd**.
4. Click the **View Report** button. You should now be able to browse the report and select a value from the report parameter lists.
5. Keep Report Manager open. You will use it in the next exercise.

**Results:** After this exercise, you should have configured the Order Details report to execute on demand and viewed the report in Report Manager.

## Exercise 3: Configuring and Viewing a Cached Report

► **Task 1: Configure the Product Sales YTD report for cached execution**

1. In Report Manager, click the **Home** link.
2. Click the **Product Sales** folder link. On the right side of the screen, click **Show Details** to view the details of each of the links.
3. Under the **Edit** column header, click the **Properties** button next to the **Product Sales YTD** report.
4. On the left side of the screen, click the **Execution** link.
5. Click **Cache a temporary copy of the report. Expire copy of report after a number of minutes**, leave the default value of 30 minutes, and then click **Apply**. Notice the error message that appears to the right of the **minutes** box, informing you that the credentials used to run this report are not stored. Because of this, the cache configuration has failed.
6. On the left side of the screen, click the **Data Sources** link. Notice that **/Product Sales/AdventureWorksDW2008** is the data source that this report uses. You cannot update the login credentials here because this data source is shared.
7. On the upper left side of the screen, click the **Product Sales** link.
8. Click the **AdventureWorksDW2008** data source link.
9. Under **Connect using**, select **Credentials stored securely in the report server**.
10. In the **User name** text box, type **NY-SQL-01\Student**.
11. In the **Password** text box, type **Pa\$\$w0rd**.
12. Select the **Use as Windows credentials when connecting to the data source** check box.
13. Ensure that the **Impersonate the authenticated user after a connection has been made to the data source** check box is not selected.

14. Click **Apply**.

**Note:** By storing credentials, you are indicating that the report should use the **NY-SQL-01\Student** account to retrieve data for all users browsing the report. Because this is a shared data source, all reports that use the **AdventureWorksDW2008** data source will run using these stored credentials. If you want to alter this behavior, you can update these reports to use custom data sources.

15. Now that you have configured the necessary database prerequisites, enable the report for caching by repeating steps 1 through 5. Notice that there is no error message and that your report is now configured for instance caching.
  16. Click the **View** tab. By viewing the report, you have created a cached instance that will expire in 30 minutes.
- **Task 2: Expire copies of the Product Sales YTD report on a schedule**
1. Click the **Properties** tab.
  2. If the execution properties are not displayed, click the **Execution** link.
  3. Click **Cache a temporary copy of the report. Expire copy of report on the following schedule** and then click **Configure**.
  4. In the **Daily Schedule** area, click **On the following days** and note that every day of the week is selected by default.
  5. In the **Start time** text box, type **11**, select the **P.M.** option, and then click **OK**.

6. On the **Execution** page, click **Apply**. In this scenario, you chose a report-specific schedule to expire the cache at the end of each day. Note that if you have several Reporting Services operations that execute at the same time, such as at the end of each day, you should consider using a shared schedule instead of a report-specific schedule.
7. Click the **View** tab and browse the report. By viewing the report, you have created a cached instance that will expire at 11:00 P.M. this evening.

**Note:** You can confirm this scheduled action by opening SQL Server Management Studio, connecting to the database engine on **NY-SQL-01**, and viewing the list of jobs in the SQL Server Agent.

**Results:** After this exercise, you should have set the cached instance of the Product Sales YTD report to expire every day at 11 P.M.

## Exercise 4: Configuring and Viewing a Snapshot Report

► **Task 1: Change the Reseller Sales report to use AdventureWorksDW2008**

1. In Report Manager, click the **Home** link.
2. Click the **Reseller Sales** folder link, and then click the **Show Details** button.
3. Under the **Edit** column header, next to the **Reseller Sales** report, click the **Properties** button.
4. Click the **Data Sources** link. To create a snapshot for this report, you must store its data source login credentials. Instead of changing the custom data source settings, you will change the report to use the **Product Sales\AdventureWorksDW2008** shared data source, for which credentials are already stored.
5. Click **A shared data source**, and then click **Browse**.
6. Expand the **Product Sales** folder, click **AdventureWorksDW2008**, click **OK**, and then click **Apply**.

► **Task 2: Schedule a snapshot to run at the beginning of the month**

1. In Report Manager, at the top right of the page, click the **Site Settings** link.
2. At the left of the screen click the **Schedules** link.
3. Click the **New Schedule** button.
4. In the **Schedule Name** text box, type **Beginning of Month**.
5. Under **Scheduled Details**, click **Month**.
6. Under **Monthly Schedule**, configure the following:
  - Select the **On calendar day(s)** option, and then type **1** to execute on the first day of the month.
  - In the **Start Time** text box, type **05:00**, and then select the **A.M.** option.
7. Click **OK**.
8. In Report Manager, on the top right of the page, click the **Home** link.
9. Click the **Reseller Sales** folder link, and then click the **Show Details** button.
10. Under the **Edit** column header, click the **Properties** button next to the **Reseller Sales** report.
11. On the left side of the screen, click the **Execution** link.
12. Select the **Render this report from a report execution snapshot** option, and then select the **Use the following schedule to create report execution snapshots** check box.
13. Select the **Shared schedule** option, and ensure that the list is set to **Beginning of Month**.
14. Clear the **Create a report snapshot when you click the Apply button on this page** check box, and then click **Apply**. The report now will not be available until it executes for the first time on the first day of the next month.

► **Task 3: Store Reseller Sales snapshots in report history**

1. On the left side of the screen, click the **History** link.
2. Select the **Store all report execution snapshots in history** check box.
3. Under **Select the number of snapshots to keep**, select **Limit the copies of report history**, and then type **12** in the text box.
4. Click **Apply**, and when asked to confirm, click **OK**.

► **Task 4: View the Reseller Sales report and report history**

1. Click the **View** tab and verify that the following error message is displayed:

The selected report is not ready for viewing. The report is still being rendered or a report snapshot is not available.  
(rsReportNotReady)

A report snapshot will only be available after a scheduled snapshot has been run.

2. Click the **Properties** tab, and on the left side of the screen, click the **Execution** link.
3. Select the **Create a report snapshot when you click the Apply button on this page** check box, and then click **Apply**. A report snapshot is created.
4. Click the **View** tab and verify that you can view the report snapshot.
5. Click the **History** tab and verify that a snapshot has been created and is now stored in history. If the list is empty, click **New Snapshot** to create the first entry.
6. Close Report Manager.
7. Shut down the virtual machine and discard changes.

**Results:** After this exercise, you should have created a snapshot history for the Reseller Sales report and viewed the report history.

# Module 7: Implementing Subscriptions

## Lab: Implementing Subscriptions

### Exercise 1: Creating a Standard Subscription

► **Task 1: Set up the lab environment**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on as **Administrator** with the password of **Pa\$\$w0rd**.
3. Click **Start**, and then click **Computer**.
4. The Computer window opens. Browse to the **E:\Mod07\Labfiles\Starter\Exercise01** folder, and then double-click **runScripts.bat**. The script executes in a console window. Wait for the console window to close before proceeding to the next step.
5. Close the **Exercise01** window.

► **Task 2: Open and deploy the AdventureWorks solution**

1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server® 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. The **Open Project** window opens. In the **Open Project** dialog box, browse to the **E:\Allfiles\Mod07\Labfiles\Starter\Exercise01\AdventureWorks** folder, click **AdventureWorks.sln**, and then click **Open**.
4. In Solution Explorer, right-click the **AdventureWorks** solution, and then click **Deploy Solution**.

**Tip:** If the Solution Explorer pane is not visible, on the **View** menu, click **Solution Explorer**.

5. In the **Output** window, confirm that all three projects were successfully deployed without warnings.
6. Close Business Intelligence Development Studio.

► **Task 3: Configure a shared data source**

1. Click **Start**, point to **All Programs**, and then click **Internet Explorer**.
2. The **Internet Explorer** window opens. In the **Address** field, type **http://NY-SQL-01/reports**, and then press **ENTER**.
3. When prompted, in the **User name** field, type **Administrator**.
4. In the **Password** field, type **Pa\$\$w0rd**, and then click **OK**.
5. If the **Information Bar** dialog appears, click **Close**.
6. Click **Product Sales**.
7. Click the **AdventureWorksDW2008** data source link.
8. Under **Connect Using**, select **Credentials stored securely in the report server**.
9. In the **User name** field, type **NY-SQL-01\Administrator**.
10. In the **Password** field, type **Pa\$\$w0rd**.

11. Select the **Use as Windows credentials when connecting to the data source** check box.
12. Ensure that the **Impersonate the authenticated user after a connection has been made to the data source** check box is not selected.
13. Click **Apply**.

**Note:** By storing credentials, you are indicating that the report should use the NY-SQL-01\Administrator account to retrieve data for all users browsing the report. Because this is a shared data source, all reports that use the AdventureWorksDW2008 data source will run using these stored credentials. If you want to alter this behavior, you can update these reports to use custom data sources.

► **Task 4: Create a standard subscription**

1. On the upper-left side of the screen, click the **Product Sales** link.
2. Click **Show Details**, and then next to the **Product Profitability** report, click the **Properties** icon.
3. Click the **Subscriptions** tab, and then click **New Subscription** to view the subscription options.
4. In the **Delivered by** drop-down list, click **Windows File Share**.
5. Verify that the **Add a file extension when the file is created** check box is selected.
6. In the **Path** text field, type **\NY-SQL-01\Reports**.
7. In the **Render Format** drop-down list, select **Excel**.
8. Under **Overwrite options**, select the **Increment file names as newer versions are added** option.

**Note:** Incrementing file names adds a number to the file suffix. Each time the subscription executes, the number increments by one.

9. Under **Credentials used to access the file share**, in the **User Name** field, type **NY-SQL-01\Administrator**.
10. In the **Subscription Processing Options**, under **Run the subscription**, ensure that the **When the scheduled report run is complete** option is selected, and then click **Select Schedule**.
11. Select the **Once** option, set the **Start time** to be three minutes later than the current time, and then click **OK**. If prompted to use AutoComplete, click **Yes**.
12. Under **Report Parameter Values**, for the **Year** and **Month** parameters, select the **Use Default** check box.
13. For the **Category** parameter, select **All Product**.
14. In the **Password** field, type **Pa\$\$w0rd**.
15. Click **OK**.

► **Task 5: View the standard subscription output**

1. Right-click **Start**, and then click **Explore** to open Windows Explorer.
2. In the **Address bar**, type **\NY-SQL-01\Reports**, and then press **ENTER**.

**Note:** Shortly after execution time that you specified above, the file should appear. This could take several minutes.

3. Double-click **Product Profitability.xls** to open it in Office Excel.
4. View the Excel output, click the **Microsoft Office Button**, and then click **Exit Excel**.
5. If you are prompted to save the file, click **No**.
6. Close Windows Explorer.

**Results:** After this exercise, you should have created and viewed the output of a standard subscription.

## Exercise 2: Creating a Data-Driven Subscription

► **Task 1: Change the Actual Vs Quota report to use a shared data source**

1. In Report Manager, click the **Home** link.
2. Click the **Territory Sales** folder link, and then click the **Actual Vs Quota report**.
3. The report requires two parameters to execute: **CalendarYear** and **Group**. When you build the subscription later in this exercise, it is important that the correct **Group** parameter value is linked to the corresponding regional **Sales Director**. This information is stored in a database table for each sales director, along with other report delivery information.
4. Click the **Properties** tab, and then click the **Data Sources** link.
5. Click the **A shared data source** option, and then click **Browse**.
6. Expand the **Product Sales** folder, and then click **AdventureWorksDW2008**.
7. Click **OK**, and then click **Apply**.

► **Task 2: View the SubscriptionGroupDirector table**

1. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, then click **SQL Server Management Studio**.
2. When prompted, ensure that **Database Engine** is selected in the **Server type** field, **NY-SQL-01** is selected in the **Server name** field, and **Windows Authentication** is selected in the **Authentication** field. Then click **Connect**.
3. In Object Explorer, expand **Databases**, expand **AdventureWorksDW2008**, and then double-click **Tables**.
4. In the table list, right-click the **SubscriptionGroupDirector** table, and then click **Select Top 1000 Rows** and view the results.
5. Close SQL Server Management Studio.

► **Task 3: Create a data-driven subscription**

**Note:** This exercise depends on credentials set in **Exercise 1, Task 3**. If you have not completed that task, please do so before beginning this task. Also, ensure that the SMTP service is running on the Virtual Machine (VM) before starting this task. Launch services.msc and verify that the Simple Mail Transfer Protocol (SMTP) service is started.

1. In Report Manager, click the **Subscriptions** tab for the **Actual Vs Quota** report, and then click **New Data-driven Subscription**.
2. In the **Description** field, type **Actual Vs Quota Subscription**.
3. In the **Specify how recipients are notified** drop-down list, click **E-mail**.
4. Select **Specify a shared data source**, and then click **Next**.
5. Expand the **Product Sales** folder, click **AdventureWorksDW2008**, and then click **Next**.
6. In the query box, type **SELECT\*FROM dbo.SubscriptionGroupDirector**.
7. Click **Validate**. Toward the bottom of the page, verify that a **Query validated successfully** message is displayed, and then click **Next**.

8. Configure the properties by using the following table:

| Property       | Option                          | Value                                      |
|----------------|---------------------------------|--------------------------------------------|
| To             | Get the value from the database | To                                         |
| Cc             | No value                        | None                                       |
| Bcc            | No value                        | None                                       |
| Reply-To       | Specify a static value          | Sales@adventure-works.com                  |
| Include Report | Get the value from the database | Include Report                             |
| Render Format  | Get the value from the database | Render Format                              |
| Priority       | Specify a static value          | Normal                                     |
| Subject        | Specify a static value          | @ReportName was executed at @ExecutionTime |
| Comment        | No value                        | None                                       |
| Include Link   | Get the value from the database | IncludeLink                                |

9. Click **Next**.
10. For the **CalendarYear** parameter, select the **Use Default** check box.
11. Under **Group**, click **Get the value from the database**, and in the drop-down list, click **GroupParameter**, and then click **Next**.
12. Select **On a schedule created for this subscription**, and then click **Next**.
13. Under **Schedule details**, click **Once**. Set the start time to execute two minutes later than the current time.
14. Click **Finish**.

► **Task 4: View the subscription notifications**

1. Wait until the time you scheduled the subscription to run, and then navigate to **C:\inetpub\mailroot\Drop**.
2. Examine the contents of the **.eml** files in the **Drop** folder.
3. Turn off **6236A-NY-SQL-01** and delete changes.

**Results:** After this exercise, you should have successfully created and viewed the output of a data-driven subscription.

# Module 8: Administering Reporting Services

## Lab: Administering Reporting Services

### Exercise 1: Using Reporting Services Configuration Manager

- ▶ **Task 1: Start the NY-SQL-01 virtual machine, log on as Administrator, and set up the lab environment**
  1. In the Lab Launcher, next to NY-SQL-01, click Launch.
  2. Log on as **Administrator** with the password of **Pa\$\$w0rd**.
  3. Click **Start** and then click **Computer**.
  4. In the **Computer** explorer windows, browse to the **E:\MOD08\Labfiles\Starter** folder, and then double-click **runScripts.bat**. The batch file 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: View Reporting Services Settings**
  1. Click **Start**, point to **All Programs**, click **Microsoft® SQL Server 2008®**, click **Configuration Tools**, and then click **Reporting Services Configuration Manager**.
  2. When prompted, click **Connect** to connect to the **MSSQLSERVER** instance on **NY-SQL-01**.
  3. View the **Report Server Status** page, and notice that the service is started.
  4. Click each item in the navigation pane in turn, and review the settings in the corresponding configuration pages.

► **Task 3: Back up the Reporting Services encryption keys**

1. In the navigation pane, click **Encryption Keys**, and then click **Backup**.
2. In the **Backup Encryption Key** dialog box, click the button to the right of the **File Location** box.
3. In the **Save As** dialog box, browse to the E:\Mod08\Labfiles\Starter folder, type **Keys** in the **File name** text box, and then click **Save**.
4. In the **Password** and **Confirm Password** boxes, type **Pa\$\$w0rd**, and then click **OK**.
5. Close Reporting Services Configuration Manager.

**Results:** After this exercise, you should have successfully viewed Reporting Services settings and backed up the Reporting Services encryption keys.

## Exercise 2: Securing a Reporting Services Site

► **Task 1: Open and deploy the Sales Reports solution**

1. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
2. On the **File** menu, point to **Open**, and then click **Project/Solution**.
3. In the **Open Project** dialog box, browse to the E:\Mod08\Labfiles\Starter folder, click **Sales Reports.sln**, and then click **Open**.
4. In Solution Explorer, right-click the **Sales Reports** solution, and then click **Deploy**.

**Tip:** If the **Solution Explorer** pane is not visible, on the **View** menu, click **Solution Explorer**.

5. In the Output window, verify that all objects in the solution were successfully deployed without warnings.
6. Close Business Intelligence Development Studio.

► **Task 2: Configure the AdventureWorks Data Source**

1. Click **Start**, click **All Programs**, then click **Internet Explorer®** and browse to <http://NY-SQL-01/reports> to open Report Manager.
2. When prompted for credentials to access the site, type **administrator** in the **User name** box and **Pa\$\$w0rd** in the **Password** box and then click **OK**.

**Note:** Do **not** enable intranet settings from the yellow Information Bar.

3. Click the **Sales Reports** folder, and then click the **AdventureWorks** data source.
4. Under **Connect using**, select **Credentials stored securely in the report server**, type **NY-SQL-01\Student** in the **User name** box, and then type **Pa\$\$w0rd** in the **Password** box.
5. Select the **Use as Windows credentials when connecting to the data source** check box, and then click **Apply**.
6. Close Internet Explorer.

► **Task 3: Open Report Manager as the NADirector user**

1. Click **Start**, click **All Programs**, then click **Internet Explorer** and browse to <http://NY-SQL-01/reports> to open Report Manager.
2. When prompted for credentials, type **NY-SQL-01\NADirector** in the **User name** box, type **Pa\$\$w0rd** in the **Password** box, and then click **OK**. Note that the user has no permissions, and can therefore see no objects in Report Manager.
3. Close Internet Explorer.

► **Task 4: Create a new role**

1. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Management Studio**.
2. In the **Connect to Server** dialog box, select **Reporting Services** in the **Server type** box, **NY-SQL-01** in the **Server name** box and **Windows Authentication** in the **Authentication** box.
3. Click **Connect**.
4. In the navigation pane on the left side, expand the **Security** item.
5. Right-click on the **System Roles** item, and then click **New System Role**.
6. In the **New System Role** dialog box, type **Security Manager** in the **Name** text box. In the **Description** text box, type **Can set item and site security**.
7. Select only the **Manage report server security** and **Manage roles** task check boxes.
8. Click **OK**.
9. Close SQL Server Management Studio.

► **Task 5: Assign the new role to the AWSalesDirectors group.**

1. Click **Start**, click **All Programs**, then click **Internet Explorer**, and then navigate to **http://NY-SQL-01/Reports**.
2. When prompted for credentials to access the site, type **administrator** in the **User name** box and **Pa\$\$w0rd** in the **Password** box and then click **OK**.
3. Click **Site Settings**.
4. In the **Security** section, click the **New Role Assignment** button.
5. In the **Group or user name** text box, type **NY-SQL-01\AWSalesDirectors**.
6. Select only the **Security Manager** role check box.
7. Click **OK**.
8. Close Internet Explorer.

► **Task 6: View site settings as the NADirector user**

1. Click **Start**, click **All Programs**, then click **Internet Explorer**, and then navigate to **http://NY-SQL-01/Reports**.
2. When prompted for credentials to access the site, type **NY-SQL-01\NADirector** in the **User name** box type **Pa\$\$w0rd** in the **Password** box, and then click **OK**. Note that the NADirector user is a member of the **AWSalesDirectors** group.
3. Click **Site Settings**. Note that the only the Security section is available.
4. Close Internet Explorer.

**Results:** After this exercise, you should have successfully created the **Security Manager** role and applied that role to the **AWSalesDirectors** security group. You should then have successfully opened the Report Manager using the **NADirector** user (which is in the **AWSalesDirectors** group).

### Exercise 3: Securing Items

► **Task 1: Assign the item permissions**

1. Click **Start**, click **All Programs**, then click **Internet Explorer**, and then navigate to **http://NY-SQL-01/Reports**.
2. When prompted for credentials to access the site, type **administrator** in the **User name** box, type **Pa\$\$w0rd** in the **Password** box and then click **OK**.
3. Click the **New Folder** button.
4. In the **Name** box, type **Samples**, and then click **OK**.
5. Click the **Properties** tab, and then click the **New Role Assignment** button.
6. In the **Group or user name** text box, type **NY-SQL-01\AWSalesDirectors**.
7. Select only the **Browser** check box, and then click **OK**. The **AWSalesDirectors** group now has permission to browse all items in the **Home** folder.
8. Click the **Contents** tab, click the **Samples** folder link, click the **Properties** tab, and then on the left side of the screen, click the **Security** link.

9. Click **Edit Item Security**, and when prompted to apply different security settings from the parent, click **OK**.
10. Select the check box next to the **NY-SQL-01\AWSalesDirectors** group, click **Delete**, and then click **OK** to confirm your action.
11. Close Internet Explorer.

► **Task 2: Browse reports as the NADirector user**

1. Click **Start**, click **All Programs**, then click **Internet Explorer**, and then navigate to <http://NY-SQL-01/Reports>.
2. When prompted for credentials to access the site, type **NY-SQL-01\NADirector** in the **User name** box type **Pa\$\$w0rd** in the **Password** box and then click **OK**. Note that the **Samples** folder link is not visible because the user has no permissions on the **Samples** folder.
3. Click the **Sales Reports** folder link, and then click the **Company Sales** report link to view the report.
4. Click the **Properties** tab. Note that you cannot set any properties for the report because the user only has permission to browse.
5. Close Internet Explorer.
6. Turn off the NY-SQL-01 Virtual Machine and discard any changes.

**Results:** After this exercise, you should have successfully created a Samples folder using Report Manager. You should have given the AWSalesDirectors security group access to all the objects under the Home directory then removed the AWSalesDirectors security group from the permissions of the Samples folder.

# Module 9: Programming Reporting Services

## Lab: Programming Reporting Services

### Exercise 1: Using URL Access to Display a Report

► **Task 1: Specify default values for a report URL**

1. In the Lab Launcher, next to 6236A-NY-SQL-01, click **Launch**.
2. Log on as **Administrator** with the password of **Pa\$\$w0rd**.
3. Click **Start**, point to **All Programs**, click **Microsoft SQL Server 2008**, and then click **SQL Server Business Intelligence Development Studio**.
4. On the **File** menu, point to **Open**, and then click **Project/Solution**.
5. In the **Open Project** dialog box, browse to **E:\Mod09\Labfiles\Starter\Exercise01**, click **ReportURL.sln**, and then click **Open**.
6. In Solution Explorer, right-click **Default.aspx**, and then click **View Designer**. Notice that there is a table located at the top of the page with seven labeled rows, and a Show Report button in the eighth row. Each labeled row contains a control permitting user input.
7. In the **Properties** window, in the list, click **ReportServerTextBox**.
8. Set the **Text** property to **http://NY-SQL-01/reportserver**.
9. In the **Properties** window, in the list, click **FolderTextBox**.
10. Set the **Text** property to **/AdventureWorks 2008 Sample Reports/**.
11. In the **Properties** window, in the list, click **ReportNameTextBox**.
12. Set the **Text** property to **Company Sales 2008**.

► **Task 2: Implement report invocations**

1. Double-click the **Show Report** button.
2. Under the **constants** comment, type the following code:

```
Const RENDER_COMMAND As String = "&rs:Command=Render"
Const TOOLBAR As String = "&rc:Toolbar="
Const ZOOM As String = "&rc:Zoom="
```

3. In the **Button1\_Click** event, under the **create report url** comment, type the following code:

```
Dim url As String
url = ReportServerTextBox.Text + "?" + _
FolderTextBox.Text + ReportNameTextBox.Text + _
RENDER_COMMAND
```

4. Under the **redirect to report page** comment, type the following code:

```
Response.Redirect(url, True)
```

5. On the **Debug** menu, click **Start Debugging**.
6. On the **Script Debugging Disabled** dialog, select the **Don't show this dialog again** checkbox, then click **Yes**.
7. In the connection dialog, log on as **Administrator** with the password of **Pa\$\$w0rd**.
8. Click **Show Report**. The **Company Sales** report is displayed in Windows® Internet Explorer®.
9. Close Internet Explorer.

► **Task 3: Configure toolbar and zoom settings**

1. In Business Intelligence Development Studio, click the **Default.aspx** tab to view the Web page in design view.
2. Click the combo box next to the **Show Toolbar** label, click the smart tag, and then click **Edit Items**.
3. In the **ListItem Collection Editor** dialog box, click **Add**, and then in the **Text** property, type **True**.

4. Click **Add**, and in the **Text** property, type **False** and then click **OK**.
5. Click the combo box next to the **Zoom** label, click the smart tag, and then click **Edit Items**.
6. In the **ListItem Collection Editor** dialog box, click **Add**, and then in the **Text** property, type **100**.
7. Click **Add**, and then in the **Text** property, type **200**, and then click **OK**.
8. Click the **Default.aspx.vb** tab to view the Microsoft® Visual Basic® code for the Web page.
9. Under the **configure toolbar and zoom** comment, type the following code.

```
url += TOOLBAR + ToolbarDropDownList.Text + _
ZOOM + ZoomDropDownList.Text
```

10. On the **Debug** menu, click **Start Debugging**.
11. In the **Show Toolbar** list, click **False**, and then in the **Zoom** list, click **200**.
12. Click the **Show Report** button.
13. In the connection dialog, log on as **Administrator** with the password of **Pa\$\$w0rd**.
14. The Company Sales report displays without a toolbar at 200% magnification.
15. Close Internet Explorer.

#### ► Task 4: Add report parameters

1. In Code view, after the **conditionally add parameter** comment, add the following code:

```
If ParameterNameTextBox.Text.Length > 0 Then
url += "&" + ParameterNameTextBox.Text + "=" + _
ParameterValueTextBox.Text
End If
```

2. On the **Debug** menu, click **Start Debugging**.
3. In the **Report Name** text box, change the value to **Employee Sales Summary 2008**.

4. In the **Parameter Name** text box, type **EmployeeID**.
5. In the **Parameter Value** text box, type **285**, and then click the **Show Report** button.
6. In the connection dialog, log on as **Administrator** with the password of **Pa\$\$w0rd**.
7. The Employee Sales Summary report is displayed for employee 285, Syed Abbas.
8. Close Internet Explorer.
9. On the **File** menu, click **Close Project**. Keep Visual Studio open—you will use it in the next exercise.

**Results:** After this exercise, you should have created an ASP.net web site that allows you to enter report parameters and open the report using those parameters. You then modified the site to include toolbar options and finally should be able to view the Employee Sales Summary report for the employee with ID 285.

## Exercise 2: Building a Report Services Web Service Client

### ► Task 1: Add a Web reference to the Report Server Web service

1. In Visual Studio, on the **File** menu, point to **Open**, and click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod09\Labfiles\Starter\Exercise02**, click **ReportManager.sln**, and then click **Open**.
3. In Solution Explorer, right-click the **ReportManager**, and then click **Add Web Reference**.
4. In the **URL** text box, type **http://NY-SQL-01/ReportServer/ReportService2005.asmx?wsdl** and then click **Go**.
5. In the connection dialog, log on as **Administrator** with the password of **Pa\$\$w0rd**.
6. After the Web service is located, in the **Web reference name** text box, change the default value (**NY-SQL-01**) to **ReportService**, and then click **Add Reference**.

► **Task 2: Add code to the form to retrieve server information**

1. In Solution Explorer, right-click **frmManage.vb**, and then click **View Code**.
2. At the very top of the file, add an **Imports** statement to the **ReportManager.ReportService** namespace, as shown in the following code:

```
Imports ReportManager.ReportService
```

3. After the existing **Inherits** statement, create a class-level variable named **rs** that references the **ReportingService2005** proxy class. Your code should appear as follows:

```
Private rs As New ReportingService2005
```

4. Locate the **frmManage\_Load** event, and before the call to the **LoadInformation** method, add the following code to set the credentials:

```
rs.Credentials = System.Net.CredentialCache.DefaultCredentials
```

5. Locate the **LoadInformation** method.
6. Modify the code as shown here:

```
Private Sub LoadInformation()
 'load information into list view
 lstView.Items.Clear()
 Dim myCatalogItems As CatalogItem()

 myCatalogItems = rs.ListChildren("/", True)

 For Each cItem As CatalogItem In myCatalogItems
 Dim strValues(3) As String
 strValues(0) = cItem.Name
 strValues(1) = cItem.Path
 strValues(2) = cItem.Type.ToString()
 strValues(3) = cItem.CreatedBy

 lstView.Items.Add(New ListViewItem(strValues))
 Next
End Sub
```

7. The code creates a list of all the items on the report server. It then loops through the list and displays the information in the list view control.

► **Task 3: Add code to retrieve individual item information**

1. Locate the **DisplayCurrentInfo** method.
2. Modify the code as shown here:

```
Private Sub DisplayCurrentInfo()
 Me.Cursor = Cursors.WaitCursor

 'display item information in messagebox
 Dim item As ListViewItem = 1stView.SelectedItems(0)
 Dim strName As String = item.SubItems(0).Text
 Dim strPath As String = item.SubItems(1).Text

 Dim properties As [Property]()
 properties = rs.GetProperties(strPath, Nothing)

 Dim sb As New System.Text.StringBuilder

 For Each prop As [Property] In properties
 sb.Append(prop.Name & ": " & prop.Value & vbCrLf)
 Next

 MessageBox.Show(sb.ToString(), _
 "Properties of " & strName)
 Me.Cursor = Cursors.Default
End Sub
```

3. The code retrieves information from the server about the item selected in the list view control. The information displays in a message box.

### ► Task 4: Test the application

1. On the **Debug** menu, click **Start Debugging**. When the form loads (this might take a few seconds), it should display the list of all available items on the report server.
2. Select one of the reports from the list view, and then click **Get Info**. Examine the properties displayed in the message box before closing it.
3. Select one of the folders from the list view, and then click **Get Info**. Examine the properties displayed in the message box before closing it.
4. Click **Close**.
5. On the **File** menu, click **Close Project**. Keep Visual Studio open—you will use it in the next exercise.

**Results:** After this exercise, you should have built a Reporting Services web service client using Visual Basic. You should know how to write a client application to consume web services from a report service and parse the information from using the service.

## Exercise 3: Using the ReportViewer Control

### ► Task 1: Add a ReportViewer control to a Web page

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to the E:\Mod09\Labfiles\Starter\Exercise03 folder, click **ReportURL.sln**, and then click **Open**.
3. In Solution Explorer, right-click **Default.aspx**, and then click **View Designer**.
4. Change the value of the **Folder** text box to **/AdventureWorks 2008 Sample Reports/**.
5. Change the value of the **Report Name** text box to **Company Sales 2008**.
6. From the **Reporting** section of the Toolbox, drag a **MicrosoftReportViewer** control onto the design surface under the existing table. Ignore the **ReportViewer Tasks** smart tag.
7. Set the **Height** property of the **Report Viewer** control to **500px**, and the **Width** property of the **Report Viewer** control to **700px**.

► **Task 2: Configure the ReportViewer Control**

1. In Solution Explorer, right-click **Default.aspx**, and then click **View Code**.
2. At the top of the **Default.aspx.vb** window, type the following code:

```
Imports Microsoft.Reporting.WebForms
```

3. Under the **use remote processing** comment, type the following code:

```
ReportViewer1.ProcessingMode = ProcessingMode.Remote
Dim serverReport As ServerReport
serverReport = ReportViewer1.ServerReport
```

4. Under the **set the report server URL and report path** comment, type the following code:

```
serverReport.ReportServerUrl = New Uri(ReportServerTextBox.Text)
serverReport.ReportPath = FolderTextBox.Text +
ReportNameTextBox.Text
```

5. On the **Debug** menu, click **Start Debugging**.
6. Click the **Show Report** button. The Company Sales report displays in the Report Viewer, beneath the parameter table.
7. Close Internet Explorer.
8. On the **Debug** menu, click **Stop Debugging**.

### ► Task 3: Add report parameters

1. In Code view, underneath the **conditionally add parameter** comment, add the following code:

```
If ParameterNameTextBox.Text.Length > 0 Then
 Dim parameter As New ReportParameter()
 parameter.Name = ParameterNameTextBox.Text
 parameter.Values.Add(ParameterValueTextBox.Text)
 'set the report parameters for the report
 Dim parameters() As ReportParameter = {parameter}
 serverReport.SetParameters(parameters)
End If
```

2. On the **Debug** menu, click **Start Debugging**.
3. In the **Report Name** text box, change the value to **Employee Sales Summary 2008**.
4. In the **Parameter Name** text box, type **EmployeeID**.
5. In the **Parameter Value** text box, type **288**, and then click the **Show Report** button. The Employee Sales Summary report is displayed for employee 288, Rachel Valdez.
6. Close Internet Explorer.
7. Close Visual Studio.
8. Turn off 6236-NY-SQL-01 and discard changes.

**Results:** After this exercise, you should have embedded the ReportViewer control in an ASP.net web page. You then should have modified the parameters of the control to allow you to view the Employee Sales Summary report.

MCT USE ONLY. STUDENT USE PROHIBITED