Joseph L. Jorden

MASTERING

Microsoft®

SQL Server™ 2005 Reporting Services Infrastructure Design

Create Meaningful Reports for Corporate Decision-Makers

Bridge the Gap Between Those Who Manage Data and Those Who Need It

SYBEX SERIOUS SKILLS.

Mastering SQL Server™ 2005 Reporting Services Infrastructure Design

Joseph Jorden

Wiley Publishing, Inc.

Mastering SQL Server™ 2005 Reporting Services Infrastructure Design

Mastering SQL Server™ 2005 Reporting Services Infrastructure Design

Joseph Jorden

Wiley Publishing, Inc.

Acquisitions Editor: Tom Cirtin
Development Editor: Dick Margulis
Technical Editor: Evangelos Petroutsos
Production Editors: Sarah Groff-Palermo and Michael Gordon
Copy Editor: Liz Welch
Production Manager: Tim Tate
Vice President and Executive Group Publisher: Richard Swadley
Vice President and Executive Publisher: Joseph B. Wikert
Vice President and Publisher: Neil Edde
Book Designers: Judy Fung and Bill Gibson
Proofreader: Ian Golder
Indexer: Ted Laux
Anniversary Logo Design: Richard Pacifico
Cover Designer: Ryan Sneed
Cover Image: © Pete Gardner/Digital Vision/Getty Images

Copyright © 2007 by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN-13: 978-0-470-11459-9

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at `http://www.wiley.com/go/permissions`.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Website may provide or recommendations it may make. Further, readers should be aware that Internet Websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (800) 762-2974, outside the U.S. at (317) 572-3993, or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data is available from the publisher.

TRADEMARKS: Wiley, the Wiley logo, and the Sybex logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Microsoft and SQL Server are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

10 9 8 7 6 5 4 3 2 1

Dear Reader

Thank you for choosing *Mastering SQL Server 2005 Reporting Services Infrastructure Design.* This book is part of a family of premium quality Sybex books, all written by outstanding authors who combine practical experience with a gift for teaching.

Sybex was founded in 1976. More than thirty years later, we're still committed to producing consistently exceptional books. With each of our titles we're working hard to set a new standard for the industry. From the paper we print on, to the authors we work with, our goal is to bring you the best books available.

I hope you see all that reflected in these pages. I'd be very interested to hear your comments and get your feedback on how we're doing. Feel free to let me know what you think about this or any other Sybex book by sending me an e-mail at `nedde.com`, or if you think you've found a technical error in this book, please visit `http://sybex.custhelp.com`. Customer feedback is critical to our efforts at Sybex.

Best regards,

Neil Edde
Vice President and Publisher
Sybex, an Imprint of Wiley

With love to my wife, Rachelle

Acknowledgments

Over the past decade or so, I've worked on a number of books and articles. None of them has been easy, but they have all been rewarding. This book is no exception; it has taken a lot of work from a lot of people who deserve a lot of praise. First, I want to thank Tom Cirtin at Sybex, for introducing me to this project to begin with, and Dick Margulis, who kept everything running smoothly after the project got under way. Special thanks to Evangelos Petroutsos, who did double-duty as the technical editor and the contributor of three chapters in the book. I would also like to thank the production editors, Sarah Groff-Palermo and Michael Gordon, copy editor Liz Welch, and proofreader Ian Golder.

Writing books takes time from personal as well as work-related activities, and I have had a number of friends help me through this book. I have to start by thanking my mother, Mary Jorden, along with the rest of my immediate family: Buddy and Shelly Jorden, and Janet, Colin, and Leian McBroom. Thanks to all of you. Also, thanks to some special people who have unwittingly taken on the task of keeping me sane through all of this (well, as close as I can come, anyway). Thanks to Jyles McDonald for the early mornings. To Dave Moody back at Jelly Belly; IOU, LE. Tristan and Amanda Williams, thanks for making me feel good about myself at pool. Most important, special thanks to my wife, Rachelle Jorden, who once again sacrificed a lot of time so that I could write another book. And finally, thanks to all of you out there for reading this book; may it serve you well.

Contents at a Glance

Introduction . . . . . *xvii*

Part 1 • Designing the Reporting Infrastructure 1

Chapter 1 • Principles of a Reporting Infrastructure 3

Chapter 2 • Determining Business Requirements 15

Chapter 3 • Planning and Developing Role-Based Security 35

Chapter 4 • Managing User Expectations 61

Chapter 5 • Change Management 75

Chapter 6 • Business Intelligence and Reporting 85

Part 2 • Using Reporting Services in Your Infrastructure 113

Chapter 7 • Deploying Reporting Services 115

Chapter 8 • Ad Hoc Reporting 139

Chapter 9 • Subscriptions 163

Chapter 10 • Maintenance and Administration 185

Part 3 • Advanced Topics 215

Chapter 11 • Managing Reports Using Program Code 217

Chapter 12 • Extending Reporting Services 245

Chapter 13 • Report Definition Language 267

Chapter 14 • Report Server Security 299

Chapter 15 • Data Sources and Data Sets 315

Chapter 16 • Performance Tuning 335

Index *463*

Contents

Introduction . . . . . *xvii*

Part 1 • Designing the Reporting Infrastructure **1**

Chapter 1 • Principles of a Reporting Infrastructure **3**
Reporting Infrastructure Basics 3
What Is a Reporting Infrastructure? 3
Do You Need a Reporting Infrastructure? 5
Why Use Reporting Services 6
What Reporting Services Can Do for You 7
Inside Reporting Services 8
Licensing Reporting Services 11
The Bottom Line 13

Chapter 2 • Determining Business Requirements **15**
Determining Current Business Requirements 16
Understanding the Current Business Model 17
Analyzing Business Processes 17
Analyzing the Company Model and Geographic Scope 20
Analyzing Organizational Structures 21
Identifying New Business Requirements 24
Gathering Accurate User Requirements 25
Identifying Key Project Stakeholders 26
Determining Business Priorities 26
Determining Risk Tolerance 27
Creating a Scope Document 27
Identifying Internal and External Dependencies 28
Creating Data-Flow Diagrams 29
Assessing Training Needs 29
Determining Regulatory Requirements 30
Defining Report Requirements 31
Planning for the Future 32
Accurate Prognostication 33
The Bottom Line 34

Chapter 3 • Planning and Developing Role-Based Security **35**
Planning Role-Based Security 35
Gathering Security Requirements 36
Setting Security Scope 37

Analyzing Your Security Requirements . . . 37
Dealing with Conflicting Requirements . . . 38
Analyzing the Costs, Benefits, and Risks of Requirements . . . 39
Implementing Role-Based Security . . . 40
Understanding Securable Items . . . 41
Understanding Tasks and Permissions . . . 41
Understanding Role Definitions . . . 44
Creating Role Assignments . . . 52
Exploring Minimum Access Permissions . . . 58
The Bottom Line . . . 59

Chapter 4 • Managing User Expectations . . . 61
Understanding Your Users . . . 62
Communicating with Your Users . . . 63
Communicate Early and Often . . . 63
Be Honest and Forthright . . . 65
Educate Your Users . . . 65
Prototyping and Storyboarding . . . 66
Training Classes . . . 68
Listen to Your Users . . . 70
Using Service-Level Agreements . . . 71
Preparing the SLA . . . 72
Writing Documentation for Users . . . 73
The Bottom Line . . . 74

Chapter 5 • Change Management . . . 75
Understanding Change-Management Theory . . . 75
Unfreezing/Changing/Refreezing . . . 76
ADKAR . . . 76
Putting Change Management into Practice . . . 77
Understanding Management's Role . . . 77
Necessary Skills for You . . . 79
Change-Management Strategies . . . 82
Managing Ongoing Change . . . 83
The Bottom Line . . . 84

Chapter 6 • Business Intelligence and Reporting . . . 85
Business Intelligence in SQL Server 2005 . . . 85
Online Analytical Processing . . . 86
Data Mining . . . 88
Installing and Configuring Analysis Services . . . 88
Installing Analysis Services . . . 89
Creating and Configuring a Project . . . 92
Creating a Cube . . . 96
Creating a Report Based on a Cube . . . 98

Creating a Report Based on a Data-Mining Model 103
Creating a Data-Mining Model 103
Creating a Report Based on a Data-Mining Model 106
The Bottom Line 111

Part 2 • Using Reporting Services in Your Infrastructure 113

Chapter 7 • Deploying Reporting Services 115
Planning a Reporting Services Deployment 116
Standard Deployments 116
Scaled-Out Deployments 120
Playing Nice with SharePoint 125
Automating Deployment 129
Automating the Installation and Configuration Steps 129
Postdeployment Configuration 132
Custom Authorization Cookies 132
Advanced E-mail Delivery Settings 133
Internet Access 135
A Deployment Checklist 136
The Bottom Line 137

Chapter 8 • Ad Hoc Reporting 139
Preparing the Systems 139
Preparing the Clients 139
Preparing the Servers 140
Working with Report Models 143
Using Model Designer 144
Using Report Manager and Management Studio 156
Using Report Builder 158
Creating and Designing Reports 158
The Bottom Line 161

Chapter 9 • Subscriptions 163
Understanding Subscriptions 164
Configuring the Servers 165
Configuring a File Share 165
Configuring E-mail Delivery 165
Configuring the Default Delivery Extension 167
Creating Subscriptions 168
Standard 169
Data Driven 172
SharePoint Integrated 179
Managing Subscriptions 180
Preloading the Report Cache 181
The Bottom Line 183

Chapter 10 • Maintenance and Administration 185
Managing the Report Server 186
Administering the Services 186
Performing Backups 189
Managing Encryption Keys 195
Managing Processing and Content 197
Report Settings 198
My Reports 199
Managing Resources 199
Managing Folders 201
Managing a Running Process 201
Troubleshooting 202
Working with Log Files 202
Tips for Common Problems 207
The Bottom Line 212

Part 3 • Advanced Topics 215

Chapter 11 • Managing Reports Using Program Code 217
Accessing Reports from within Your Code 217
Accessing Reports with a URL 218
Viewing Reports 218
Using the *rs:Command* Parameter 220
The *rs* Utility 221
Using the *rc* Parameters 223
Passing Report Parameters in a URL 224
The Report Server Web Service 225
Accessing the Report Server from within *.NET* Applications 226
Selected Management Methods 231
Selected Execution Methods 236
Specifying Device Information Settings 240
The Bottom Line 243

Chapter 12 • Extending Reporting Services 245
Understanding Extension Types 246
Building a Security Extension 246
The *IAuthenticationExtension* Interface 247
The *IAuthorizationExtension* Interface 250
Deploying a Custom Security Extension 254
Building a Data Processing Extension 258
The *IDbCommand* Interface 259
The *IDbConnection* Interface 260
The *IDataReader* Interface 261
Deploying a Data Processing Interface 263
Deploying a Data Processing Extension to Report Designer 264
The Bottom Line 265

Chapter 13 • Report Definition Language 267
What's in an RDL File 267
The Structure of an RDL File 269
Creating a Report Definition File 272
Deploying the New Report 279
Specifying Report Parameters 280
Creating Reports in Code 281
Embedding Images 287
Exploring the RDL Elements 289
The *Report* Element 289
The *EmbeddedImage* Element 289
The *ReportParameters* Element 290
The *ReportParameter* Element 290
The *ParameterValue* Element 290
The *DataSet* Element 290
The *Fields* Element 290
The *Field* Element 291
The *Query* Element 291
The *QueryParameters* Element 291
The *QueryParameter* Element 291
The *DataSources* Element 291
The *DataSource* Element 291
The *Body* Element 292
The *PageHeader* Element 292
The *PageFooter* Element 292
The *Table* Element 293
The *TableColumns* Element 293
The *TableColumn* Element 293
The *Header* Element 293
The *Footer* Element 293
The *TableGroups* Element 294
The *TableGroup* Element 294
The *Details* Element 294
The *TableRows* Element 294
The *TableRow* Element 294
The *TableCells* Element 294
The *TableCell* Element 295
The *Textbox* Element 295
The *Image* Element 295
The *Style* Element 295
Global Variables 296
The Bottom Line 296

Chapter 14 • Report Server Security 299
Authentication 299
Types of Authentication 300

Configuring Report Server for SSL 301
Changing Service Accounts and Passwords 303
Authorization 305
Securing Folders 307
Securing Models 308
Securing My Reports 311
Preventing SQL Injection Attacks 312
The Bottom Line 313

Chapter 15 • Data Sources and Data Sets 315
Defining Report Data Sources 315
Creating a Shared Data Source 317
Using Data Source Expressions 321
Defining Report Data Sets 321
SQL Server Relational Database Data Sets 322
Analysis Services Data Sets 323
SQL Server Integration Services Data Sets 327
XML Data Sets 328
Oracle Relational Database Data Sets 329
SAP Relational Database and NetWeaver BI System Data Sets 330
Hyperion Essbase Data Sets 332
Report Model Data Sets 332
The Bottom Line 333

Chapter 16 • Performance Tuning 335
Using the Performance Console 335
Modifying Configuration Files 343
ReportingServicesService 343
RSReportDesigner 343
RSWebApplication 346
RSReportServer 348
Tips and Techniques 356
Setting a Measurement Baseline 356
Data Archiving and Trend Tracking 356
Manually Configuring Available Memory 356
The Bottom Line 357

Appendix A • 359

Appendix B • SharePoint Integration 391

Appendix C • Report Server Web Service Reference 417

Index *463*

Introduction

Reporting Services was introduced with SQL Server 2000, and it quickly gained popularity because it was easy to work with and it came free with SQL Server 2000.

A lot has changed since its initial release over seven years ago. Several new features make Reporting Services easier to use and more functional:

- Reporting Services now allows users to create their own reports using the Report Builder application.
- Developers can create reports, report models, and other SQL Server–related projects using the new Business Intelligence Development Studio, which is based on the familiar Visual Studio interface.
- There are several new tools for administrators to use in configuring the report server, such as SQL Server Reporting Services Configuration Tool, the SQL Server Surface Area Configuration Tool, and SQL Server Management Studio.

With all of the changes, a lot has stayed the same. For instance, Report Manager is still the primary interface for end users and administrators alike, and it looks almost the same in this version.

The Report Server is still separated into two separate services: the Windows service and the Web service. The Windows service still provides encryption and scheduling services, and the Web service still handles ad hoc requests and serves up the API to developers.

And here's the really good news: everything you've written in the previous version of Reporting Services will still work in the new version. All of your reports will still render, your subscriptions will still run, your custom extensions will still function — everything will still work after the upgrade.

There is a lot more to Reporting Services than just this basic list, though, and you will be exposed to the various functions and procedures in this book. Whether you've worked with Reporting Services previously or this is your first time, you are going to be impressed with what this new version can do.

How this Book is Organized

We've designed this book to be a reference for the user new to Reporting Services or the experienced user who might want to see what's new in this version. The emphasis is not just on getting up and running, but on making sure that you do it right the first time. We don't try to cover every nook and cranny of the system, because that would take a much larger book. We have, instead, provided you with the essential information you will need to get oriented with the product and get it running in your environment.

The book is divided into three parts:

Part 1 (Chapters 1–6) Part 1 is designed to help you craft a project plan for deploying Reporting Services in your enterprise. You will learn what makes a reporting infrastructure, how to determine business requirements for the new reporting system, and how to manage change during the deployment. There is also a section on role-based security so that you can plan the security for your new system before you even install it.

Part 2 (Chapters 7–10) Part 2 helps you to take all of the theory from Part 1 and put it into practice. In Part 2 you will learn how to deploy your reporting infrastructure based on the plans you've drawn up in Part 1. You will learn how to manage report models for users to create ad hoc reports and how to create subscriptions. There is also a chapter on managing and troubleshooting Reporting Services.

Part 3 (Chapters 11–16) Part 3 digs into the more advanced Reporting Services topics. You will learn how to program for Reporting Services by accessing the API. You will also get an in-depth view of the XML language that reports are made from, the Report Definition Language. There are also discussions on security, data sources and data sets, and Report Server maintenance.

How to Contact the Author

As the old saying goes, nobody's perfect. So, while we have tried to make this book as accurate as possible, there will inevitably be the occasional mistake. There will be updates, service packs, and release versions that change this software. If something strikes you as odd, or you find an error in the book, please drop me a line via e-mail at `jljorden@comcast.net`; I'm always happy to hear from my readers.

Part 1

Designing the Reporting Infrastructure

When most of us get a new piece of software, we are anxious to install it right away and start playing with it. With Reporting Services, it is best to quell that inclination because we have a lot of preparation work to do first.

In this part, we will discuss what you need to do before installing the software so that you get a usable reporting infrastructure that everyone in the company can be happy with.

There are six chapters in this section:

- **Chapter 1: Principles of a Reporting Infrastructure**
- **Chapter 2: Determining Business Requirements**
- **Chapter 3: Planning and Developing Role-Based Security**
- **Chapter 4: Managing User Expectations**
- **Chapter 5: Change Management**
- **Chapter 6: Business Intelligence and Reporting Services**

Chapter 1

Principles of a Reporting Infrastructure

Businesses can't be run without some kind of reporting system. Management needs to know what is going on in the company in order to make accurate business decisions, so they need reports. Because you are reading this book, it is obvious that you are considering Reporting Services to get those reports out.

Before you can make a solid decision, you must know what you are looking into. What is Reporting Services and what can it do for you from a business perspective?

In this chapter we will:

- Define a reporting infrastructure
- Figure out why an enterprise-reporting infrastructure is necessary
- Help you decide if you need to use Reporting Services as your reporting infrastructure

Reporting Infrastructure Basics

Before you can decide whether to spend all of the time and money necessary to construct a reporting infrastructure, you need to know what one is and how it is used.

What Is a Reporting Infrastructure?

This sounds like a simple question, but it is more involved than it seems. You may think of a reporting infrastructure as any old type of reporting system that your business may use — from pen and paper to colossal database systems. But there is more to it than that.

A reporting infrastructure is not only the data source but also the mechanism that displays the report in a human-readable format, as shown in Figure 1.1. For instance, if you keep a grocery list in your head and you need someone to go to the store for you to pick some things up, you write down a list of what you need. The data source is the list you keep in your head, and the pen and paper are the mechanism used to display the report in human-readable format.

Of course, most modern reporting infrastructures are considerably more complex than that. Businesses use a variety of systems, depending on the data they keep and their reporting needs. Some may use a simple spreadsheet to tally sales at the end of the day, some may use a Microsoft Access database with custom reports, and others may use an enterprise-reporting system. That is where Reporting Services starts to make sense.

An enterprise-reporting infrastructure like Reporting Services is used to display information from disparate data sources in a format that people can understand and use in making business decisions. What does that mean? Let's consider an example.

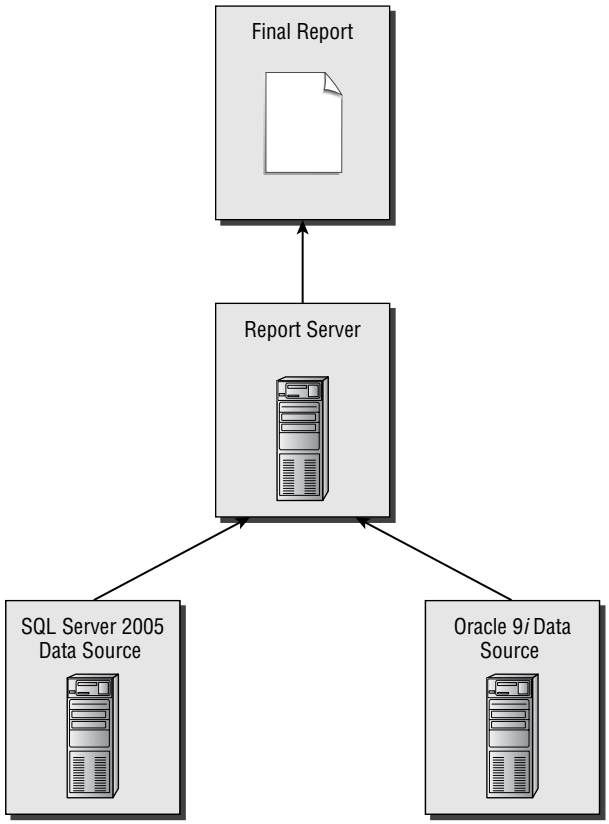


FIGURE 1.1
An enterprise-reporting infrastructure is used to combine data from different sources into a human-readable format.

Most large companies have a number of databases that contain data for different departments and different purposes. Human Resources may have a database for keeping track of employee records, the Maintenance department may have a database for keeping track of equipment, and so on.

Management needs to keep track of which employees have what equipment so that when someone gets promoted they can get new equipment needed for their new position. Or, if someone gets terminated, management can get all of the equipment back from that employee.

Maintenance does not have any records of who gets promoted or terminated in their database, and Human Resources does not keep track of equipment in their database, so by themselves these databases are not useful for tracking equipment based on employee status.

However, if you use an enterprise-reporting system, you can retrieve data from both of these data sources and combine them into a single report that helps management see which employees have what equipment and what needs to be done with that equipment.

Now let's go one step further. Human Resources may have a system that uses Microsoft SQL Server 2005 for data storage and Maintenance may use Oracle 9*i* for storing data. No problem. An enterprise-reporting system can combine data not only from different data sources but also from different vendors. That is what a reporting infrastructure is used for, but do you need one?

Do You Need a Reporting Infrastructure?

The short answer to this question is yes, you do. The real question is: What kind of reporting infrastructure do you need to put in place? To answer that question you need to consider a few basic things:

- How much data do you need to report on?
- Where is that data stored?
- How many reports do you need?
- How often do you need them?
- How complex are those reports?

Your answers to these questions will dictate whether or not you need an enterprise-reporting system. For example, if you have a small spreadsheet of weekly sales figures and you just need a total of the sales every week, you don't really need an enterprise-reporting infrastructure. In fact, it would be a waste of money in this case.

If, however, you have several database servers with multiple databases that you need to combine into multiple reports on a regular basis, then it is prudent for you to consider an enterprise-reporting infrastructure.

The most common business scenario is somewhere in between these two extremes. Many companies have a single database server that houses data for all of their departments in separate databases. They may have other small, external data sources, like spreadsheets, to report on as well. If your company falls into this category, you should consider an enterprise-reporting structure.

Take our employee/equipment report, for example. This report was based on sizable databases in two different departments that are stored on two different database systems. It would be difficult and costly to use a simple reporting tool to generate any meaningful reports from these two data sources. Sure, it could be done using something like Access or FileMaker, but these tools were not designed with this capability in mind. These applications were designed to report on small, local databases. So getting them to talk to large, remote databases is a time-consuming, costly task. And the reports are usually slow when they are complete.

In contrast, an enterprise-reporting infrastructure is designed specifically for combining data from multiple data sources, provided by different vendors, into reports that can be used by anyone. Because the enterprise-reporting system is designed specifically for this task, the reports are generated quickly, especially in comparison with the nonenterprise counterparts.

So if you have a lot of data on disparate systems, you definitely need an enterprise-reporting system to generate custom reports quickly. Now you just need to know why you should use Reporting Services to fill this need.

Real World Scenario

Case Study: Talksalot Communications

Talksalot (not their real name) were doing a lot of business with some of the big-name vendors in the communications market. They sold everything from home phone service to large business communication services like T1 lines.

Every month Talksalot received commissions from their vendors, along with accounting reports. The reports came in two formats — paper and electronic (on magnetic tape). The personnel in the company did not know how to use mag tape, so they tried to reconcile their commission checks using the paper copies of the accounting reports. This took so much time that they often did not complete the task before the next round of reports came in.

A consultant came in and wrote a small application to control a mag tape reader and read the tapes automatically. All the users had to do was load the tape. The application read the tape and transferred the data into a database for storage. The consultant then wrote a number of reports that showed revenue and credits for the different avenues of business. Talksalot was able to review the accounting reports from their vendors in less than a day.

What to Look For in a Reporting Infrastructure

There are several features that you and your users will likely expect from any enterprise-reporting infrastructure, so look for these features when making a decision:

- Quality reports — No matter how complex a report may be, users want them to be formatted in a professional manner that is easy to read.
- Personalization — Not all users have the same needs, so you should be able to deliver reports to the user in a format they can use with content they need.
- Localization — If you are in an international organization, you need to be able to display reports to users in their native language and cultural context.
- Interactivity — Your users should be able to navigate existing content easily and create new content on the fly.
- Private branding — Reports should blend seamlessly into existing applications and websites. End users should not know that they are looking at a report generated by a separate system.
- Extensible content tools — You need to be able to extend the infrastructure to include new data sources and reporting capabilities that may not be included out of the box.
- Browser-based report delivery — Everybody has a web browser on their computer these days, so you need to be able to deliver reports to the browser.
- High performance — The infrastructure has to be as fast as possible because users do not want to wait for their reports. They want their data now.
- Manageability — You need to be able to manage all aspects of the infrastructure easily, so make sure that it comes with the proper management tools and APIs.
- Scalability — When your company grows, you need to be certain that your reporting infrastructure can grow with it.

Why Use Reporting Services

We've established that you need a reporting infrastructure of some kind. More complex reporting needs require a more robust reporting infrastructure. Why should you use Reporting Services as a reporting infrastructure, though? What can it do for you?

Although these may seem like trivial questions to the technically inclined, they are very important questions from a business perspective. This is especially true when you consider the expense involved in installing and configuring a reporting infrastructure.

Consider this from your boss's perspective. There is a lot involved in getting a reporting infrastructure up and running in a production environment:

- You have to purchase the software.
- Support personnel have to be trained.
- Developers will need training.
- The software will need to be installed across the enterprise.
- The software will then need to be configured and tuned.
- Developers will need to write new reports, plug-ins, supporting database objects, etc.

This can be an expensive undertaking that would be difficult, if not impossible, to back out of once you get started. So it is important to understand what Reporting Services can do for your organization and how it does so.

What Reporting Services Can Do for You

While it is true that Reporting Services is an enterprise-reporting infrastructure that can combine data from a variety of data sources into a report that you can make some sense out of, there is more to it than that.

Reporting Services supports the entire reporting lifecycle, beginning to end. Here is what the lifecycle entails:

Report Authoring One of the most important tasks that a reporting infrastructure must support is creating new reports. In Reporting Services, developers can use any tool that supports the Reporting Definition Language (RDL), which is an XML-based language used to define reports. Of course, Reporting Services has some basic tools for creating new reports built right in.

Report Management Once developed and released, reports can be accessed on a scheduled basis or on demand. The reports are usually cached on disk and an execution history is kept. Reporting Services has built-in tools for keeping track of the execution history, managing the cache, scheduling report delivery, and performing other management tasks. You can also access management functions via web services if you want to incorporate these functions into your own applications.

Report Delivery Users can receive reports in one of two ways. The report can be pushed to the user based on an event (a push report), or the user can request a report specifically (a pull report). The users can either get the report from a web page, which they can then print or save as a Microsoft Excel spreadsheet or a PDF, or they can have the report e-mailed to them.

Report Security You do not want everyone in the company to have access to every report available, so you can secure your reports. Reporting Services implements a role-based security model that you can extend to use custom security models if necessary.

These capabilities allow you to design reports for a number of different reporting scenarios:

Enterprise Reporting This is the primary function of Reporting Services. Developers can design reports at any location, on any supported data source, and deploy them to a central

location for users anywhere in the organization to access. For example, a developer in California can develop a report and deploy it to a server in Denver for a user in Bangkok to access.

Ad Hoc Reporting SQL Server 2005 Reporting Services comes with a fancy new tool called Report Builder, which allows users to create their own reports. This can be handy for users who need a report for that 8:00 meeting when the developers do not get in until 9:00.

Web-Based Reporting Everyone in business today has to work with other businesses to some extent, which means that they need to share data. Because Reporting Services delivers reports via the Web, it is an excellent tool for sharing data with vendors over the Internet or extranet.

All of this makes Reporting Services an excellent choice for your reporting infrastructure from a business perspective. But you are probably wondering how this works. So let's get a quick overview of how Reporting Services does its job.

Inside Reporting Services

SQL Server Reporting Services is a collection of several components. The main component is the Report Server. We call it the main component because this is the workhorse that processes incoming report requests and makes reports available to end users. Further, the Report Server is actually two different components combined into one: a Windows service and a web service. Let's see what these two main components do.

Report Server Components

The Windows service provides the necessary framework for getting Reporting Services up and running, and keeping it that way. The Windows service initializes the instance of Reporting Services, making sure the necessary databases exist and are ready for use, ensuring that all of the service components are available, etc. This service provides scheduling and delivery services for getting reports to end users. It also provides maintenance functions to keep the services running at peak performance.

The web service exposes a set of programmatic interfaces that developers can use to access the Report Server from their custom applications. Quite a few commands are available to you as a developer that you can access through your application, which makes the web service a very powerful tool.

These two components combine to form the Report Server, but there is still more. The Report Server does its work by using a number of subcomponents.

Report Server Subcomponents

The Report Server is made up of several subcomponents, as seen in Figure 1.2. Some of the subcomponents are divided into smaller components still. Let's look at an overview of what these subcomponents are and what they do.

Processors The Report Server actually uses two processors to perform the tasks of preparing and delivering reports. These cannot be modified or extended by developers.

Report Processor When a user makes a request for a report, this processor retrieves the report definition from the Report Server database, initializes any parameters and variables in the report, and performs some preliminary processing to get the report ready for delivery.

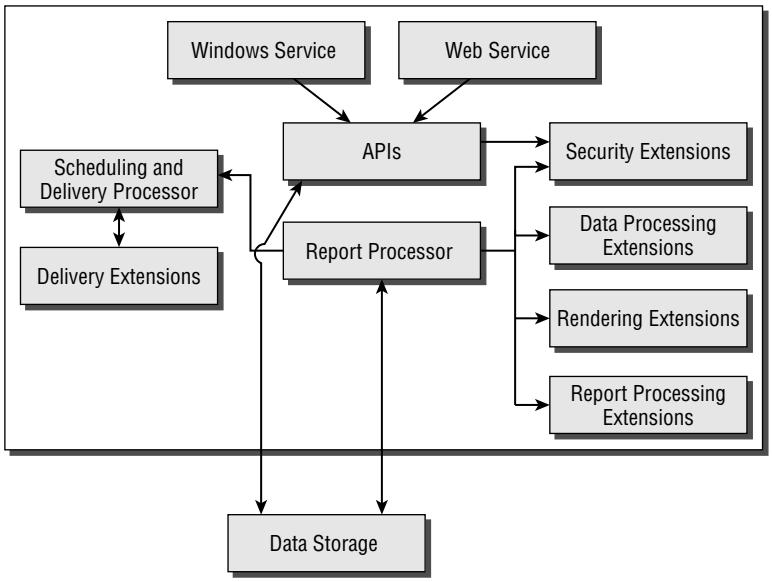


FIGURE 1.2
Reporting Services comprises several components and subcomponents.

Scheduling and Delivery Processor Reports can be scheduled for delivery to e-mail inboxes and shared folder destinations. This processor handles that scheduling functionality. To do this, the component maintains a queue of scheduled events and notifications. When an event fires, this component calls the report processor to execute a report, process a subscription, or perform some scheduled maintenance. After the report is generated, the scheduling processor calls the appropriate delivery extension to get the report to the user.

Data Storage Everything that users see in Reporting Services is stored in a database. The database contains metadata, report definitions, and the folder hierarchy displayed in the Report Manager.

Extensions Extensions are used to perform a variety of functions, such as rendering reports and processing data. These differ from processors in that developers can create new extensions to meet their own needs.

Security Extensions Security extensions are used to authenticate and authorize users who try to access reports on the server. The default method is Windows Authentication. If that does not meet your needs, you can create a custom security extension. A custom extension cannot run alongside the default extension because there can only be one security extension in place at a time.

Data Processing Extensions When a report is processed, this extension type is used to query the underlying data source and return a flattened rowset. The basic tasks needed to accomplish this are as follows:

1. Open a connection to the data source.
2. Analyze the query and return a list of field names.

3. Run the query against the data source and return a rowset.
4. Pass any required parameters to the query.
5. Iterate through the rowset and retrieve data.

There are several types of data processing extensions available right out of the box, including SQL Server, Oracle, OLE DB, ODBC, and Analysis Services. If these do not meet your needs, you can write your own. Unlike with security extensions, you can have multiple data processing extensions running side by side.

Rendering Extensions These extensions transform data from the Report Processor into a format that users can read and understand. There are six of these extensions by default, and you can add more as you see fit. Let's look at the default rendering extensions.

HTML Whenever a report is requested through a web browser, this extension is used to render the report. All HTML from this extension is generated using UTF-8 encoding.

Excel This extension is used to generate reports that can be viewed and changed using Microsoft Excel 97 or later. This extension generates files in the native file format for Excel, called the Binary Interchange File Format (BIFF).

CSV The CSV rendering extension creates plain-text files that have values separated by commas. There is no formatting applied. CSV files can be opened using most spreadsheet applications as well as standard text processor applications.

XML Like the name implies, this extension renders reports into an Extensible Markup Language (XML) format, which can then be used in other applications. This extension can also use an Extensible Stylesheet Language Transformations (XSLT) schema to transform the resultant XML into any schema you may need. All of the XML is generated using UTF-8 encoding.

Image This extension can render reports in a variety of image formats, including TIFF, BMP, EMF, GIF, JPEG, and PNG. This is very useful in a few scenarios. Using this extension ensures that the resultant report looks the same on every client, no matter what application they use to view it. This extension can also be used to send a report directly to a printer or fax machine for transmission to users.

PDF This extension renders reports in a PDF format that can be read by Adobe Acrobat 6 and later.

Report Processing Extensions Reporting Services features a variety of items that can be included in your reports for processing and displaying data. Out of the box you can include tables, charts, matrices, text boxes, images, and more. However, if you want to include something more exotic, such as a custom map from a mapping application, you would need to create a custom report processing extension. You can have many of these installed on a single server at the same time.

Delivery Extensions These extensions are used to deliver reports to their destination so users can access them. There are two of these by default.

E-mail The e-mail delivery extension is used to send reports to users via Simple Mail Transport Protocol (SMTP). It can send the entire report in the body of the e-mail, or just a URL with a link to the report on a web server. Short notices without the link or

report can even be sent to Short Message Service (SMS) devices like pagers and cell phones.

File Share As you might expect, this extension delivers reports to a file share on a network drive that users can access. You can specify the location, rendering format, filename, and overwrite options to control this extension.

Now that you know what Reporting Services can do for you and how it does what you need, all you have to know is how much it costs. The actual price changes from vendor to vendor, so we can't discuss specific prices here. We can, however, help you understand the licensing requirements.

Licensing Reporting Services

Before you can make a valid business decision to start using Reporting Services in your organization, you need to know how much the software is going to cost. You should contact your vendor about specific prices, but if you understand the licensing model you will be better equipped to make a decision.

The good news is, SQL Server 2005 comes with the license to run Reporting Services. You do not need to buy a separate license to run the software or have users access it. There is a caveat, though. You need to have a valid SQL Server 2005 license for the system that reporting services is running on. There are three scenarios to consider when looking at licensing.

Scenario 1: Single-Server Environment If you run SQL Server 2005 and Reporting Services on the same computer, you do not need to purchase any extra licenses, as illustrated in Figure 1.3. You are fully licensed at that point.

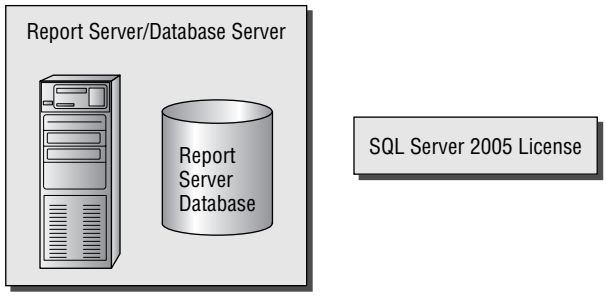


FIGURE 1.3 Only one license is required for a single-server environment.

Scenario 2: Multiple-Server Environment If you need to run SQL Server 2005 and Reporting Services on separate computers, as shown in Figure 1.4, you will need to purchase a SQL Server 2005 license for both systems.

Scenario 3: Web Farm A web farm, from a Reporting Services perspective, is a number of computers that have Reporting Services installed and that all connect to a separate server that has a single Reporting Services database shared by all instances, as illustrated in Figure 1.5. In this case you will need to purchase a SQL Server 2005 license for all of the front-end systems (the ones running Reporting Services) and a license for the database server.

Now all you need to do is decide which environment you need to run and purchase the correct number of SQL Server licenses and you will be in business.

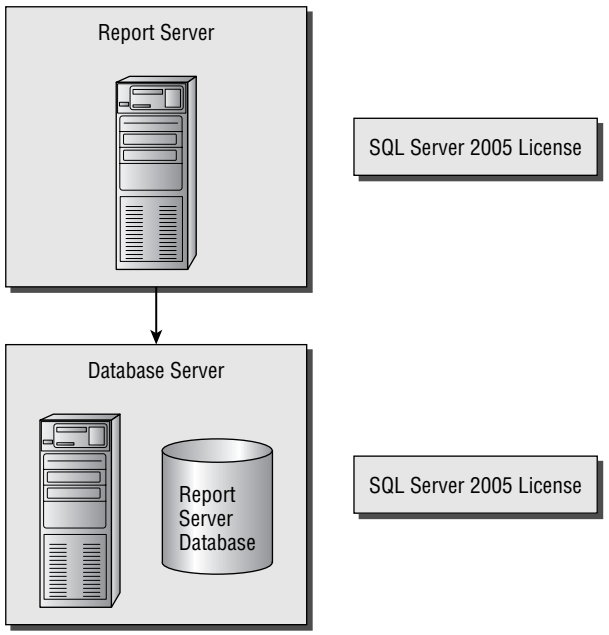


FIGURE 1.4
You will need a license for all servers in a multiple-server environment.

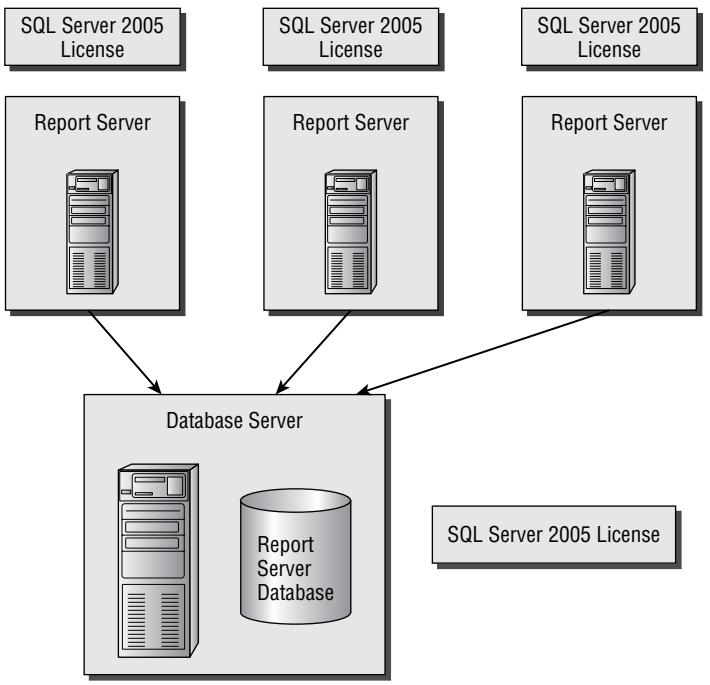


FIGURE 1.5
You will need a license for each system running Reporting Services and the database server in a web farm.

The Bottom Line

Define a reporting infrastructure. Simply put, a reporting infrastructure combines one or more data sources with a mechanism for displaying the data in a format that makes sense to end users.

Master It What are the two primary components of a reporting infrastructure?

Figure out why an enterprise-reporting infrastructure is necessary. Just because you have data does not mean that you need an enterprise-reporting infrastructure. You may only have a small amount of data that you report on infrequently, and the reports may be fairly simple. This scenario may not require an enterprise-reporting infrastructure. If you have a lot of data from a number of data sources that you require complex reports from, then you should seriously consider an enterprise-reporting infrastructure.

Master It You work for a mid-sized company that has a Human Resources database on an Oracle 9*i* system and an Accounting database on SQL Server 2005. You need to generate salary and bonus reports for upper management based on both of these databases. Does this require an enterprise-reporting infrastructure?

Decide if you need to use Reporting Services as your reporting infrastructure. Reporting Services can easily handle any reporting task you can throw at it, so functionality is not the question here. The real question is how much Reporting Services will cost to implement. Talk with your vendor for actual pricing, but you will find that, with the right licensing, Reporting Services is a definite contender.

Master It You work for a mid-sized company that has an inventory management database on a SQL Server 2005 server. You need to create a report for your customers that shows your current inventory. Your customers need to access this report over the Internet. Can you use Reporting Services to meet your needs?

Chapter 2

Determining Business Requirements

Let's face it: the whole reason that most of us IT professionals got into our trade is because we like to work with new technology. Sometimes the enticement is so great that we forget that there are business issues that drive the need for technology and we forge ahead, undaunted. This approach, while great fun, is completely backwards.

Before implementing any technology, including Reporting Services, you need to define the business needs that drive the technology. Think about the problems that you are trying to solve before you start to solve them. When you consider implementing Reporting Services, there is only one real business need to take into account: people have data that they want compiled into a report that is easy to work with.

This may seem like a simple business problem, but you need to answer a few questions to fully comprehend the issue. For example, why does the user want the data compiled into a report? The most common reason is to see a simple management report, like an employee headcount report for Human Resources. Other users may need reports for specific purposes, such as government audits, payroll status, and specially formatted packaging labels. In any of these cases, a reporting infrastructure is a good solution to the business need.

Once you decide to implement Reporting Services, you are probably inclined to just start developing reports and see how it goes. Unfortunately, there is still more to do from a business perspective. You need to figure out how you will use each of the Reporting Services elements and why. Getting the answers to these questions will help:

What is the purpose of the reporting infrastructure? If you don't know why you are implementing a reporting infrastructure, you will not be able to do it right. So, before you start, make certain you understand everything you are being asked to do.

Who are the subject matter experts? Every report has a subject: accounting, sales, production, all of them. Each of these subjects has a subject matter expert, someone who knows the report data inside and out, backward and forward. And, as much as it may pain you to admit it, you are almost never that expert. You need to find those experts to help develop and confirm the accuracy of new reports.

What types of users and applications will access these reports? In order to design useful reports, you need to know who will be using them. If you have a technically unskilled user base, you should make the reports as simple as possible while still containing enough data to be useful. You need to know if your users have access to e-mail and network systems so you will know if you can send reports via e-mail or file share access. The more you know about your users, the more you can do to design an effective reporting infrastructure.

What are your plans for the future? As your business needs evolve, your users' needs will change. Reports that were helpful six months ago may be useless today. Make sure that your implementation can change and expand with the needs of the company.

As you can see from this brief overview, there is a lot to consider before you start implementing a reporting infrastructure. If you don't factor in the business needs that drive the technology, you will end up with a reporting system that does not meet the company's needs. However, if you spend some time hashing out the business requirements, the result will be a reporting infrastructure that you can be proud of.

For the remainder of this chapter, you are being promoted to manager. You will need to think like a manager and answer questions that affect the entire enterprise. Once you have answers to these questions, you will be able to use Reporting Services to help achieve your company's goals.

In this chapter you will learn to:

- Determine your current business requirements
- Analyze the current business model
- Identify new business requirements
- Plan for future business requirements

Determining Current Business Requirements

Imagine that you are standing at the bank of a river that you need to cross. There is no bridge, so you need to build one. The first thing you do is assess what materials are available around you; wood, rope, tools, etc. Then you scan the other side of the river to see where to connect the bridge to, looking for the safest location. After you determine all of this, you can start building a bridge that will take you safely across the gap.

Determining business requirements is a similar process. You first need to assess your current standing, determining what is available to you now. Then you need to clearly define where the business needs to be at the end of the project. Only then can you build a bridge that will take you safely across the business gap.

Most businesses have a reporting infrastructure of some kind in place already. This makes it even more important to assess your business's current standing because you have to design a system that interacts with an existing database system or reporting infrastructure. This first step consists of two phases: gathering requirements and applying those requirements to the system.

The requirements-gathering phase is a challenging and important process. To illustrate why this is so, let's return to our bridge-building scenario. Imagine that you start building your bridge and halfway through, you realize that you don't have enough wood to complete it. Because you didn't assess your current situation well enough, the construction phase is stalled and you will never make it to the other side. If you had used less wood from the start, tried a different design, or used other materials, you may have been successful, but now you will never know.

That is what makes the requirements-gathering phase so important. If you do not assess what tools, materials, and needs you have at the start of the project, you will have problems later on — guaranteed.

The key section of this phase is the process of determining the business requirements, which describe the business reasons for implementing or upgrading a computer system. If you think of the bridge illustration again, this is like asking why you need to cross the river in the first place.

To determine the business requirements, you need to have a thorough understanding of the current business state, which requires you to understand how all of the key functions of your organization, including Reporting Services, are working together to achieve business goals such as higher sales or lower operating costs. Using this data, you can start to construct a model of

what the new system should look like and what it should be able to help you accomplish. This data is the starting point for building your bridge.

So put on your hard hat — it's time to start building.

Understanding the Current Business Model

To start bridging the gap between where you are and where you need to be, you must have a basic understanding of the business model that your organization uses. This is a key factor because having this insight will help you see what reporting components are in place already and how you can use those components to your advantage. At the very least, you should include some of the existing reporting capabilities in your new infrastructure because your users have come to rely on them for performing their daily functions.

As you review this business model, you will gain a deeper understanding of the industry that your organization is part of. This is great for you because this insight will help you to anticipate your customers' needs and, in some cases, wants. So take the time to gain an understanding of the business model by interviewing key stakeholders in the project — listen to them, and take notes. Some of what they have to say may be a bit dry, but it is all very important to you because every bit of information is going to help you design a better solution. If the users think they had a hand in this solution, they are going to appreciate it more.

Also, consider another important factor, one that is often forgotten. Many times when a large project is undertaken, upper management looks at it as an opportunity to change the way they do business in other areas. Sometimes they make a multitude of smaller changes; other times they make massive changes. Either way, you should be prepared to accommodate these changes in your project to help the company achieve its business goals.

This is also a good time to determine which of the existing reports are still valuable. Many of the reports that your company currently has in its reporting system were developed for a good reason. Over time, though, business needs change, people come and go, and reports lose their value. Unfortunately, many of the reports that have outlived their usefulness are still in the system. You do not want to waste valuable time trying to migrate these old reports to your new system, so attempt to figure out which of the current reports are actually still in use.

Most important, though, is the buy-in factor. Users need to feel as if they are being heard. If you just go about implementing a reporting infrastructure with little or no user input, the users will not be happy and you will get nothing but complaints about the system no matter how great it is. If you gather business requirements from your users and actually listen to what they need, they will love your system and defend it to the bitter end, even if it has some flaws that must be ironed out. Bear in mind that you may not use all of your users' suggestions, if any. If you take the time to understand your users' needs, and let them know why some of their requirements did not make it into the final product, they will be much happier with the end result. That makes requirements gathering a very important activity. It all starts by analyzing business processes.

Analyzing Business Processes

To analyze the current business state and really understand where the business is at now, you need to know what the business does and how it does it. This involves analyzing the current business processes.

A *business process* is a set of work activities designed to produce a specific output for a particular customer or market. For example, think of what it takes to make a shirt:

1. The business needs to buy cloth and buttons.
2. They need to have employees cut and sew the cloth based on a pattern.

3. They need to sell the shirt to a retail store.
4. They need to deliver the shirt.
5. They need to collect the money for the shirt that was sold.

It is actually more complex than that, but you get the idea. There is a process for everything that happens in an organization. If you have a good grasp of those processes, you will understand the goals of the organization better. Once you understand the goals of the organization, you can design your reporting system to help the company reach those goals.

During your analysis you will also find out who needs access to what data and how they need that data to be formatted. This is essential in designing a useful reporting infrastructure, so take it seriously.

While you are analyzing the current business processes, compare them to the anticipated needs of the company. You may find that some of the current processes can be improved, or even done away with altogether.

Reengineering Business Processes

There are many companies around today that came up with their business processes years ago, when the market or economy was different. Some companies developed their processes as they were growing and the processes came together out of necessity. Some don't even realize that they have processes and find out during a large-scale project that these processes are in place. So, after analyzing all of the current business processes, some companies decide that it is time for a change and they reengineer their business processes.

Often, a reporting infrastructure project is not the driving force behind a reengineering project but a result of it. While analyzing their existing processes, many companies find that they have inadequate reporting systems in place and decide to implement a new system.

Companies take several actions when reengineering business processes:

- Comprehensively review the current business processes.
- Achieve an overall understanding of the business operations.
- Identify all tasks that are part of the business processes.
- Review tasks to identify those that reengineering can improve.
- Develop strategies to improve the effectiveness of these processes.

There is no definitive list of all the business processes in use today, but they generally fall into one of four groups: information flow, communication flow, service and product lifecycles, and decision making.

Information Flow

Information-flow processes define how information is distributed throughout the company. Such processes define what information is available, who needs the information, and how they get it. It is important to understand this type of process because it has a direct effect on your reporting system.

When you think about it, a reporting system is well designed to support an information-flow process. When you define what information is available, you have defined the data source for

the report. You can then define who needs access to the data and give them access, and you can define how they get the information by delivering the report in a format they can read. A good understanding of the information flow processes in your organization allows you to better define your reporting infrastructure.

Communication Flow

While an information-flow process defines where data needs to go, a communication flow process defines how the information gets where it is going. This type of process defines how information flows through your business systems, including your network, during the course of day-to-day business operations.

Understanding the communication flow processes in your organization will help you to identify the performance characteristics of your current reporting systems. This is important because you want to implement a system that is just as fast, if not faster, than your existing reporting system. One way to make the new system faster is to look for ways to improve the communications flow while you are analyzing the systems already in place.

Analyzing the communications flow of your existing system requires you to identify the existing data sources and destinations. Bear in mind that this is not always an electronic format. The data source may not be a database; it may be a logbook that truck drivers sign when picking up shipments. You may be able to modify these data sources to use an electronic format and include them in your reporting infrastructure. Also, remember that the data destinations are not always people. Sometimes the destination is an unattended e-mail inbox or a custom application that processes data. Be sure to document these destinations during your analysis.

The documentation that you generate during your analysis should include a data-flow diagram (which we'll discuss a little later in this chapter). This is a visual representation of where the data originates, where it ends up, and all of the communication steps in between. This is a useful tool because it can help you visualize your data transmission and help you find areas that may require attention.

Service and Product Lifecycles

Many products and services for sale in the market today have a finite life span. For instance, when you buy a car you buy a model based on the year it was made (for example, a 2007 Nissan Pathfinder). In 2008, the 2007 models are no longer offered for sale on new-car lots. The period of time that a product or service is offered to the public is called the product's *life span*.

When analyzing business processes, look at how your company comes up with new products and services and gets them to market. This process, from start to finish, is referred to as the *lifecycle*. There are several aspects of the lifecycle that you should understand:

- Conceptualization
- Research and development (most companies simply call this R&D)
- Design
- Prototyping
- Testing
- Production
- Customer support
- Retirement

The product's life span is shorter than its lifecycle. The life span is how long a product is available to the public. The lifecycle accounts for the period of time that the product is being developed, the time it is available to the public (the life span), and the period of time it is supported after it is taken off of the market.

To fully understand the lifecycle for your company's products and services, you need the answers to a few questions:

- How fast is the lifecycle? Does a product move quickly from concept to retirement, or is it a slow process?
- How long is the product life span: days, months, years?
- How does each aspect of the lifecycle affect your reporting infrastructure?

The answers to each of these questions can have a bearing on the reports that you create. For example, if you create a report to show weekly sales of a new product, you will not need that report after the product has gone off the market. If you have a product that has reached the end of its life span but is still being supported, you may need to create a report to show how many support calls you receive for that product.

Decision Making

Businesses don't become successful by accident. Someone made a series of conscious decisions to get the business where it is today in the industry. In business, even the act of making decisions is based on a process. Some of these processes are faster than others, some require a great deal of supporting documentation and approval, and most fall somewhere in between.

You need to incorporate these processes into your design. The decision-making process affects how your project is run and how long it will take. It also affects how users are approved for gaining access to data and how new data sources are integrated into a reporting system. So you need to understand the decision-making processes at your company in order to successfully implement a new reporting system.

There is more to it than how your project and report systems are affected, though. You have to think about the company. Remember that the reports generated by your new system will be used in making business decisions that can affect the profitability of your organization. You must ensure that the reports used in these decisions are delivered quickly to the right people and that they are accurate and easy to understand.

Analyzing the Company Model and Geographic Scope

The company model and geographic scope affect your reporting system more than you may think. Going back to our illustration, think of how the geography affects the bridge you are trying to build to cross a river. If you are trying to cross a river that is at the bottom of a deep gorge, you would probably build a suspension bridge. If you are trying to cross a river that is at ground level, though, a suspension bridge would not be feasible.

The same is true of your reporting infrastructure. You need to consider how the company's physical layout and model affect your system. You may run into a few common models:

International Many large organizations operate in more than one country, which makes them an international organization. This is the most complex model because it has more issues that can arise and that you must consider. For example, you need to determine what languages are used across the organization, what cultures are included, how common pieces of data are formatted (for example, dates), and so on.

National This model is a little less complex than the international model because it applies to a company that spans only a single nation. You do not usually need to worry about language barriers and cultural issues. You do need to concern yourself with the complexities of data access, support across time zones, and any regional or branch office concerns.

Regional The regional model is a subset of the national model, spanning a small section of a country. For example, in the United States you may have the Western region, which could encompass all of the states west of the Rocky Mountains. This is less complex than the national model because you only need to concern yourself with issues in a specific region.

Branch Branches are small offices that are part of a larger organization, usually national or international. Only a few people work in these offices, and they need the data that is available to the rest of the company. This is a challenge for you because you have to figure out what data they need in each branch and how to get that data to the users.

Subsidiary A subsidiary is a company in and of itself, but it is owned by another company (called the parent company). The biggest challenge in a subsidiary is usually company politics. You have to determine who needs access to data in the parent company and how to get them that access both from a technical and a political standpoint.

Analyzing Organizational Structures

To thoroughly understand the current state of the business, you need to be familiar with the organizational structure. The organizational structure defines the roles and duties of each person in the company and shows where everyone fits in relation to the company's goals and objectives.

It is important that you have a good grasp of the organizational structure when implementing a new reporting system because this structure defines who should have access to reports generated by your system. It is also important to know who has authority to grant access to those reports.

Ideally, a good organizational structure should meet four primary objectives:

- Provide a method for allocating tasks, roles, and responsibilities
- Coordinate efforts and tasks among departments and cross-functional teams
- Establish a clear reporting structure
- Provide a clear path for distribution of information

Organizational structures are best understood by reading an organizational chart, which is a visual representation of the hierarchy of the organization. As seen in Figure 2.1, these charts usually look like flowcharts, with supervisors at the top and direct reports underneath. These charts are excellent tools for illustrating how the business is organized and maintained.

Keep in mind that organizational charts are not infallible. They may be outdated, or they may imply a formality that does not exist in the company. Be sure to double-check that you have the latest and most accurate information when reading the chart.

Understanding the Current Structure Model

When you are analyzing the company's current structure, you need to know what type of structure it is. There are a few common types that it may fall under:

Functional A company that employs the functional organizational structure is organized around skills and departmental duties. This works well in smaller businesses where business processes are well defined.

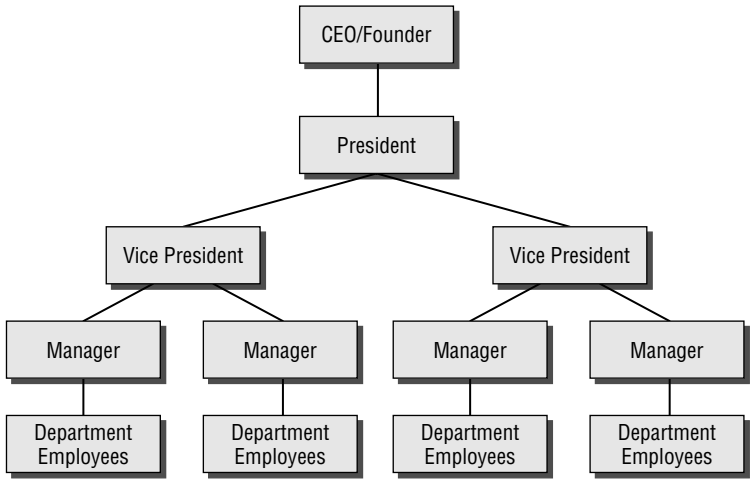


FIGURE 2.1
An organizational chart can help you visualize the company's management structure.

Matrix In the matrix structure, the company is organized by function rather than by department. For example, suppose your company makes clothing, such as pants, shirts, dresses, and suits. In a matrix structure you would not work in the IT department; you would work for the Men's Clothing function, or the Suit Manufacturing function. While it may look unwieldy on paper, this structure allows flexibility and speedy response when executed properly.

Geographic This organizational structure works well for international and national companies that are trying to target different products for different areas. For instance, a sporting goods company may want to market snowmobiles in a colder region and surfboards in a tropical region.

Decentralized Business Units In this structure, each business unit operates almost like a separate company, called a profit center. This is a good structure for companies that have completely separate lines of products and services to market. For example, a food service company may have a chain of restaurants that sells Chinese food and another chain that sells Mexican food, each operated as a separate profit center.

These organizational structures do not fit every business; in fact they are almost never a perfect fit. More often than not, you will find that your company is using a hybrid of two or more of these types.

The primary reason for analyzing the current business structure is that the current structure usually defines who gets reports and what type of reports they need. This will help you define who should get reports with the new system. For instance, when going over the organizational structure you may find that upper management demands certain reports that are not made available to middle management. You may also find that some reports are required in one geographic location and are useless in another. All of this is valuable information for you to have before implementing your reporting system.

To get a full understanding of the company's organizational structure, you must look beyond the organizational chart. Go out and talk to the people who make the company function. Find out what they do and how they are organized. You will most likely find alliances and agreements that are not part of the organizational chart but that still have a profound impact on who receives which reports.

Several key factors can affect the shape of organizational structures:

- Management model
- Company organization
- Vendor, partner, and customer relationships
- Company politics

Management Model

The first thing you need to define when looking into the organizational structure is the management model. Specifically, what you need to determine is whether this is a private or a public company.

If this is a privately owned company, then the founder is usually the primary stockholder. This can allow the company to respond quickly to market changes because decisions are usually made by only a few people. However, some obviously ill-conceived ideas can be put into place because they were not ratified by a group.

If this is a public company, then the founder is not the primary stockholder and there is usually a board of directors that sets policy for the company. This model has good and bad points as well. It can be good because decisions usually require input from the entire board of directors, so more thought is typically put into the decisions. On the downside, decisions can take longer to make, which slows down the business.

No matter which management model the business follows, there are a few questions that you need to answer:

- Is the management style bureaucratic or authoritarian?
- Does management stress accounting and close control?
- Does management encourage people to be creative and take initiative?
- Is management willing to take risks, or do they prefer to ''stay the course''?

Having the answers to these questions will help you devise a proposal that is more likely to be accepted by management.

Company Organization

The way your company is physically arranged will have a major impact on the design of your reporting infrastructure, so you must take it into account. This is so because you will have to make sure that your resources are distributed appropriately among the users that need them. Consider some of the ways that a company may be physically organized:

- Different departments in separate sections of a single floor
- Different departments on separate floors of a single building
- Different departments in separate buildings on a single campus
- Different departments in separate buildings in different sites

If the entire company fits in a single building then you will not have to worry too much about distributing resources. If, however, your users are dispersed across multiple geographic regions, then you will have to ensure that the reporting resources are readily available to everyone who needs them.

Vendor, Partner, and Customer Relationships

In times past, companies provided their vendors with invoices and receipts on paper and if the vendor requested anything else, they didn't get it. Nowadays businesses are starting to allow the vendors, partners, and customers access to a variety of information beyond simple receipts and invoices.

You may need to factor this in when designing your reporting infrastructure. If your company allows external users to access reports, a good strategy is to incorporate these reports into an Internet or extranet site to provide access.

Company Politics

If you have been working with computers and networks for a while, then you have probably heard of the Open Systems Interconnection (OSI) model, which defines the seven layers of networking protocols. The often-overlooked eighth layer of the OSI model is politics, because company politics can affect your systems just as much as network cards and operating systems can. How so?

First, people want power, and when a large change is undertaken they see it as an opportunity to gain more power. They may feel that if they, or their department, has ownership of a certain report, they will have more power. This may sound petty, but some people will actually lobby and petition management for this control until they get it. Like it or not, this affects your project.

You should account for people resisting the changes that you are going to make with the new reporting system. Plan on dealing with these political maneuvers by adding some padding for them in your timeline.

Also, plan on making some concessions for those people who want more control for themselves or who want to delegate some of the extra work they already have. Planning ahead for this political rivalry will help you come up with a more accurate and reliable timeline.

Identifying New Business Requirements

Now that you have an inventory of your current systems, the next step is defining what the business needs to gain from a new reporting infrastructure by determining the future business requirements.

You can identify these future needs by answering questions like these:

- How does management see the business working in the future?
- What are the business problems that management wants to solve by implementing a new reporting system?
- Will the company need to make any changes to the existing organizational structure? If so, what will they be?
- Can you integrate any of the existing business processes into the new design?
- Will you need to create new business processes to accommodate the new design?

There are several tasks that you will need to focus on to help you answer these questions:

- Gathering accurate user requirements
- Identifying key project stakeholders
- Determining business priorities

- Determining risk tolerance
- Creating a vision/scope document
- Identifying internal and external dependencies
- Creating data-flow diagrams
- Assessing training needs
- Determining regulatory requirements
- Defining report requirements

Gathering Accurate User Requirements

While most businesspeople know how to use a computer to get their daily work done, most of them are not technically savvy. That makes sense because not everyone works in the technology field. Accountants, maintenance workers, shipping managers — all are necessary to make the business run smoothly but they do not know the ins and outs of the technical systems.

This presents a challenge for you. Your users know what they need to see in their reports, but they do not know how to explain what they need in technical terms. That means that you must be able to translate the users' requirements into technical documentation that systems administrators and developers can use to create a reporting infrastructure that actually meets those user needs. To translate what the users are asking for, you need to understand how users think about computers versus how IT personnel think about computers.

We IT folk think of computers along technical lines and we describe their behavior in terms of the components they use. For instance, if we need to create a report we may talk about adding a matrix control to a report to connect to a data warehouse cube.

Users do not think of computers that way. Nontechnical people do not think in terms of capabilities and components; they think in terms of behavior. A user knows what the system currently does and what they want it to do. If they want a report with a matrix control to connect to a data warehouse, they might say something like, ''I need a report that shows me the sales data for the current month and I would like it to look kind of like a spreadsheet because the lines make it easier for me to read.'' That is the kind of stuff you will be translating into requirements for your technical personnel.

You will also need to temper the users' expectations at this point so that they are not disappointed later. If your users ask for a reporting system that has a telepathic interface and delivers reports through a mind link, then you need to let them know that the system is limited in that way. Of course, you won't (hopefully) get such a request, but some users do ask for things that are not technically feasible, or even possible, and it will be your job to let them know that the system may not support some certain requirements.

The main point here is to make absolutely sure that you understand what the users need, and that they understand what the system is capable of doing. Then you can take these requirements back to your technical personnel and help them turn the user requirements into a functional reporting system.

This process may take several iterations. You may have to write out a rough draft of the requirements and review them with the users to ensure that everyone is in agreement. The first time you review the documentation, I guarantee there will be some changes. The second time, and possibly even subsequent times, there will be changes to the specification. Don't despair, though; this is normal. All of this effort leads to a better system that everyone is happier with.

Identifying Key Project Stakeholders

To ensure that the reporting system is going to help the company achieve its goals, you must identify the people who set those goals and the people who help the company meet those goals. These people are the key project stakeholders.

The first person to contact is the project sponsor. This person has a major stake in the project, usually staking a promotion or pay raise on the outcome of the project. They have been given the authority to manage the project and make important decisions about how the project will proceed, what the budget for the project is, and whether or not the project should even be permitted to continue after a certain point. So it is important for you to find this person and make sure that you understand what they want out of a reporting system and that they understand what you can deliver to them.

Next, find the users of the current business systems. These people can tell you what the current system is capable of and where it is lacking. You can then use this information to make sure that existing functionality is ported to the new system and try to fill any missing gaps in existing functionality.

Other key information sources are:

- Existing IT personnel
- Company legal staff
- Corporate management
- Customers (when relevant)

Determining Business Priorities

You can't please all of the people all of the time. That is especially true when implementing a reporting infrastructure. Some people will want features that other users just cannot work with; some users will want reports that other users do not want published. You will have to determine what actually ends up making it into the final implementation, essentially pleasing as many of the people as possible. How do you do that?

You will have to start by determining the business priorities, which are based on business goals set by management. So set up a meeting, or series of meetings, with the senior management to find out what these business goals are. Take good notes during these meetings so that you can make a list of these goals.

Once you have a comprehensive list of business goals, sort through them and figure out which ones apply to the reporting infrastructure process, because not all of them will apply.

The next step is to prioritize these goals. To do this effectively, you need input from management and the user community to find out what they feel is important, what they need to do their job, and what they can live without. Then, using that information, assign a number to each business goal, 1 being the most important, 2 being less important, and so on. You do not need to assign a unique number to each goal; you may have two or three goals that are priority 1, for example. This will help you see what needs to be included in the project and what can be left out. Figure 2.2 shows what this list may look like.

FIGURE 2.2
Make a list of your project's goals and prioritize each goal.

Priority	Goal
1	Make reports available to overseas users
1	Make reports available to branch offices
4	Generate reports in color
3	Put company logo on all reports
2	Generate manufacturing reports every Friday at 10:00

Sometimes you will find that some people disagree with the assigned priorities. They may feel that their priority, which was labeled as a 4, should have been labeled at least a 2. This is where all of those notes come in from the meetings with senior management. When everyone understands what the business goals really are, they may be more willing to compromise.

The key is to make sure that the majority of those involved understand the priority assigned to the project goals and agree with your assessment. Otherwise, no one will be happy with the outcome of the project.

Determining Risk Tolerance

Think back to our bridge analogy for a moment. When you start scouting for a place on the far shore of the river to connect your bridge, you have to consider the risks involved in getting there. How deep is the water where you plan to build the bridge? Is the ground on the other side stable or too muddy and soft? You must factor in these risks in order to safely build your bridge.

The same is true of a reporting system project. You have to consider the risks involved. It is true that no one is going to lose their life over these risks, but a great deal of time and money may be lost that could do damage to the company as a whole. So you must determine the tolerance for risk in your project.

The first thing you need to do is conduct a risk assessment for your project. There are three steps to this process:

1. Determine the risks involved.
2. Determine the magnitude of the potential loss.
3. Determine the probability of the loss.

To assess the risks involved, you will use the information you have gathered from end users and management about how they use the current system. Some of the reports that are currently in use are necessary for day-to-day operations, so business may slow down or stop altogether without these crucial reports. Other reports are not as important and can be offline for a longer period of time.

Once you determine what the risks are, the next step is to identify the impact of these risks on the business. If your shipping department uses the reporting system to print bills of lading (BOL), then shipping may stop if the BOL report is not available. This can cost the company hundreds of dollars per hour. Your job here is to find out exactly how much the company will lose in terms of productivity, sales, or money if a crucial report is offline.

The last step is to figure out the percentage of probability of the risk happening. If there is only a 2 percent likelihood of the risk occurring, then you do not have to account for it as seriously as if there were a 75 percent chance.

Once you have determined these three factors, you can better understand which risks are acceptable and which are not. Then you can plan for these risks in your project. For example, it is an unacceptable risk for the new reporting system to fail on startup, so you should plan on leaving the old one in place until sometime after you have started the new one to account for that risk.

Creating a Scope Document

The scope document is one of the most important documents you will generate during the entire project. This document serves a number of important functions:

- It is used to make sure that all of the decision makers and stakeholders involved in the project have a clear understanding of the goals for the project.
- It lays out the risks and alternatives involved in the project.

- It identifies the resource requirements for the project.
- It is used to present a budget for approval.

It is important that this document be as complete and accurate as possible, so make sure that you include reasons behind each conclusion. Think of this as showing your work on a math problem. Your high school math teacher did not want to see just the answer; he wanted to see the reasoning you used to get there. Managers and stakeholders do not want to see just your conclusions; they need to see your line of reasoning to make sure that the expenses are properly justified.

Because it is so detailed, this document is usually quite lengthy and it can take a long time to produce. In fact, it may take several weeks to create a proper scope document, so allow plenty of time for this in your project plan. And do not cut corners on this document; it is going to be used to justify the budget for the entire project.

Identifying Internal and External Dependencies

To successfully complete your reporting system project, you will need to rely on people to help you and on the completion of other projects. These other things that you rely on to complete your project are called *dependencies* and there are two types: internal and external.

An *internal dependency* is something internal to your project that can affect the outcome of your project. For example, you need to create test reports before you can call your project a success, but you cannot create test reports until you have installed Reporting Services. The act of installing Reporting Services is therefore deemed an internal dependency.

Most internal dependencies involve resource availability. People on your project may get sick, take vacation, or resign. Computer systems involved with your project may crash or need to undergo periodic maintenance that requires downtime. To accommodate these types of issues you may need to assign alternate resources. If a key person is ill for an extended period, assign another person to fill in for them. In case a computer system crashes, make sure you have a backup server in reserve to compensate.

An *external dependency* is something that is outside of the scope of your project that can affect its outcome. For instance, you need to install the Reporting Services software on a server to start working with it, but you may not be able to install the software until the new servers come in from the vendor. The servers being delivered by the vendor is an external dependency because you have no real control over the vendor's schedule.

External dependencies are more challenging to deal with because you inherently have little or no control over them. For example, if your vendor is several days late on delivering a new server for you to install your software on, there is little you can do to speed them up. You cannot send them more people to help or buy materials for them to use to make your new server faster. The best thing to do is add some extra time to your project for such issues that may arise to give yourself a safety cushion.

Bear in mind that it is not necessary to document and track every single dependency. That would involve hundreds, and possibly thousands, of dependencies. You do not need to track each and every person on the project; you can get by with tracking groups of people as dependencies. You may not need to track each vendor or each computer system separately either, they can be tracked as groups too. This will help you simplify your project plan while still keeping track of anything that can affect the schedule.

This task is much easier if you use project management software, like Microsoft Project.

Creating Data-Flow Diagrams

A data-flow diagram is a graphical representation of the way data flows through an information system — for instance, how data flows from a custom application to a database and then out to a report.

The system relies on data flowing from one entity to the next in order to display it as a report. What may not be clear is what should be included in the diagram and how it should be labeled. Several symbols are used in a typical data-flow diagram, and each has a special meaning:

- A square represents an external entity.
- A circle represents a process.
- Arrows are used to represent data flows.
- Two parallel lines that look like an open-ended rectangle represent a data store.

An *external entity* is anything outside of your system that affects data flow. For example, if you have a website where customers can place orders for product, then the end user's computer would be an external entity that could be represented by a square or an oval.

A *process* is anything that happens on the system that accepts input and works with it in some way. If you have a website that customers use to order product, then you probably have an application the automatically downloads those orders from the website and prepares them for shipping. This custom application is a process and would be represented by a circle or rounded rectangle.

The process of moving data from one source to another is called a *data flow*. For example, when the data is moved from your website to your custom download application, the data is said to flow from the website to the application. This would be represented by an arrow pointing to the custom download application.

Of course, every reporting system has a data store somewhere. It does not need to be an electronic store, but for Reporting Services it is always electronic, usually a database.

Once you have an accurate data flow of the existing system, you are better equipped to design the data flow for the new system. You can use the existing data flow to make sure you do not lose functionality with the new system and you can look for improvements to incorporate.

There are a number of software packages that can help you create a data flow diagram, such as Microsoft Visio.1.3

Assessing Training Needs

Whenever you implement a new technology, your users will require some amount of training. One of the tasks you need to undertake is finding out how much training is required to get your users up and running on the new system as fast as possible.

To determine how much training is required, you need to develop a thorough training plan, which should involve everything necessary to provide valuable education.

To start, determine who needs what education. As an example, your end users in the Manufacturing department probably do not need training on how to install the software on a new server, but your system administrators in the IT department do. When you find out what type of training your users need, you need to determine how much is needed. Remember that some users are more advanced than others and do not require as much training to get up and running on a new system.

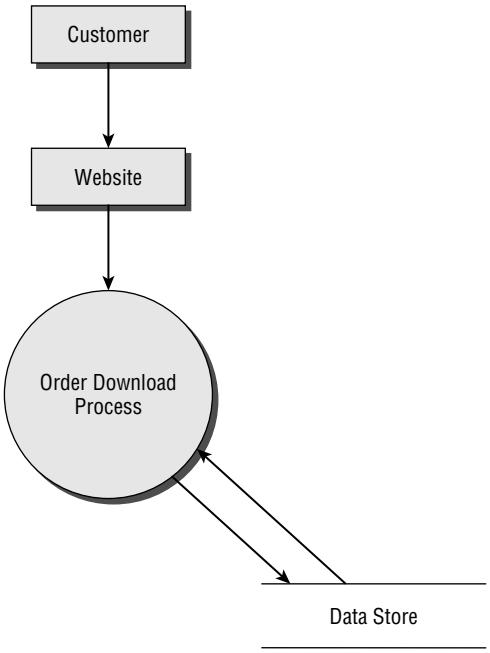


FIGURE 2.3
A data-flow diagram helps you visualize how data traverses your information systems.

Once you know what kind of training is needed, and how much, the next step is deciding where the training should be provided. You may be able to find a training provider that offers courses on the software, which is the case for Reporting Services. You may find that the training is not available in your area, or that it does not meet your specific needs. In that case you may have to create your own specialized curriculum and deliver it on-site.

If you decide to create and provide your own training, bear in mind that this is a time-consuming and costly process. You should retain the services of a professional trainer who has experience in developing custom curriculum and have that person teach the courses for you. Unless you are experienced at training and developing curriculum, do not try this yourself. It is more difficult than it seems.

Determining Regulatory Requirements

In addition to the internal business requirements set out by your own management group, your company may be subject to regulations imposed by external groups, such as government agencies or insurance companies. You need to take these into account when creating new reports and data sources for your reporting infrastructure.

All businesses are affected by external regulatory bodies to some degree. For instance, all businesses in the United States have to record their income and pay taxes to the Internal Revenue Service. So while all businesses are affected, they are not all affected to the same degree. Your job is to understand the degree to which these regulatory bodies affect your organization's reporting needs.

A good example of a company that might be affected is a food producer. The Food and Drug Administration (FDA) requires that food be labeled with specific information on the package. The FDA requires specific nutritional data in a specific format. It even dictates the size of the

label based on the size of the package. If your company makes food products, then you need to know what is required to print legally acceptable labels through your reporting system.

Of course, you are probably not a lawyer, so it is not reasonable to expect you to understand all of the laws governing your business. That is why it is important for you to seek out the company's legal counsel and listen to their advice. They can help you to avoid legal pitfalls that can make your project a disaster.

If you are working for an international organization, this task becomes more of a challenge. Not only do you have to obtain legal counsel from lawyers in your own country, you also have to apply legal advice from other countries where your organization operates. This is not as difficult as it sounds, though, because you do not need to get legal advice for every single report that you process, but only for those that are obviously governed by an outside agency. Take a list of the reports that you are going to create to your legal counsel or regulatory affairs office and ask them which ones you should pursue.

Real World Scenario

CASE STUDY: CHOCOLATY GOODNESS CANDIES

Chocolaty Goodness Candies makes candies and confections that are sold and enjoyed all over the globe. This is great for business — and challenging for reporting.

Some of the challenges faced by the Reporting Services team involved labeling the various packages. In the United States, the FDA mandates that certain packages have labels of a certain size. Each of these labels requires specific information, such as sodium content, carbohydrates, grams of fat, and so forth. This information has to be listed in a specific order as well.

What makes these packages even more challenging to work with is the fact that the company sells the candies in other countries. Australia, for example, requires a different size of label for a 2-pound bag than the United States requires for the same-size bag. Australia also requires different information in a different order.

Without the help of legal counsel and a few phone calls to the proper people in the proper countries, the Reporting Services team would not have been able to generate legal labels for the packages — and the company would not have been able to sell its fine product. The legal counsel really helped in this case.

Defining Report Requirements

Going back to our bridge analogy, once you have taken an inventory of what you have on hand and decided where you need the bridge to go, you must determine what kind of bridge to build. If you have a lot of wood and rope handy, then you will probably build a primitive suspension bridge. However, if you happen to have a team of welders and riveters with an ample supply of steel girders, you may build a modern suspension bridge.

With your reporting infrastructure project, once you know what you have on hand and you have determined what management and end users expect from your new reporting infrastructure, you can start defining the components that will make up the new system, including:

- How many servers are necessary and where they will be placed on the network
- What reports will be created

- What components are required to create those reports
- Who will have access to the reports
- How the reports will be delivered

You can accomplish this by answering some basic questions about the infrastructure and each report, such as:

- Infrastructure questions
  - How many sites are there, and how many users at each site?
  - Does each site need its own server, or can they access another server over the network?
  - Are the data stores for the reports local? If not, can the network handle the offered load of traffic?
  - If the reports are to be e-mailed or saved on a file share, are these network resources available to everyone?
- Report definition questions
  - Who is the report being designed for?
  - Who should have access to the report?
  - Who determines whether the report meets the needs of the business?
  - How will the report be delivered?
  - How will users obtain the report? Do they request it or is it scheduled?

This is not a comprehensive list of questions, but just enough to get you thinking so you can come up with a list of your own. Once you have the answers to these questions, you are almost done. There is just one thing left: forecasting future needs.

Planning for the Future

There is an old saying that goes, ''If you fail to plan, then you plan to fail.'' This is true of any project that you undertake. If you are building a bridge to cross a river, you will have to plan for storms and flooding. If you do not, then a flood could easily collapse your bridge because you did not take it into account. If you do not plan for the future in your reporting project, your system can become useless in a short period of time. How so? Take the Sarbanes-Oxley Act as an example.

You probably remember hearing about the various companies that engaged in shady and downright criminal business dealings around 2000. Well, in 2002 the U.S. government passed the Sarbanes-Oxley Act in an attempt to curb illegal activities in large corporations.

Some of the provisions of this new law required greater disclosure of business dealings. Greater disclosure means that more information needs to be collected and shared with government agencies and stockholders. In turn, that means that many companies had to make changes to their reporting infrastructure.

If these companies could not make the necessary changes fast enough, they would be fined hefty sums of money until they could comply. Eventually, a company could go out of business if it was unable to comply with these new legal requirements.

Those companies that planned for the future were able to make the needed changes much faster than companies that planned only for their present needs.

Of course, this is just one example, and it is one of the more flamboyant ones at that. More common reasons to change a reporting infrastructure are:

- Hiring more employees (or laying them off)
- Moving into new markets
- Creating new product lines
- Opening new offices
- Changing from a national to an international organization
- Allowing vendors and partners access to reports over an extranet

Think about it; suppose that you do not currently allow your vendors access to your reports over an extranet. Then, in a few months your management staff informs you that you will lose one of your biggest customers if you do not allow them extranet access to some of your reports. Imagine how relieved you will be if all you need to do is flip a switch to grant them access. You will not be able to do that so easily if you do not plan for the future.

So, when you are designing your new reporting infrastructure, think about what you need now and what you might need later. Review your business goals to see if there is the possibility of buying another company, going international, allowing vendors access over an extranet, and the like. Than make sure that your design can handle these possible future needs as well as definite current needs.

Accurate Prognostication

It is not possible for us to know the future until it has become the past, so accurate prognostication may seem like an oxymoron. It is possible for you to compensate for that lack of knowledge in your project plans so that you can be better prepared for future changes.

Suppose that one of your company's business goals is to add 50 percent revenue over the next four years and that they expect a 20 percent increase in the number of employees to accomplish this goal. If your company currently employs 200 people, then you should plan for another 40 people using the system within the next four years. That seems simple enough, but it is almost never accurate.

It is may be reasonable for your company to plan to increase revenue by 50 percent over the next four years, but what if they only increase revenue by 10 percent due to a changing economy? What if they hire new salespeople and manage to increase by 75 percent? In either case the company's needs will be different than projected.

To compensate for these changes, do not implement them right at the start. What that means is, do not buy the extra hardware and software to accommodate the growth until after it has occurred. Plan for the anticipated growth in your long-term budget, but wait until the right time to spend the money. If your company is making more money than anticipated, then management will not be too upset about giving you more money for more servers and software.

If your company ends up making less than expected, management will be happy that you already have what you need and then some. Either way, everyone involved will be happier with the outcome.

To ensure that you make projections everyone will be happy with, here are a few things you should keep in mind:

Projections are assumptions, not facts. A projection is a guess about the future, which is an unknown variable until after it has come to pass. Make sure that your project plan has enough flexibility to survive bad assumptions. In other words, plan for the future, but do not act on it until it comes.

Involve the right people. You will not be able to make any projections about the business on your own. You will need input from the business management and stakeholders to help you make these projections. Talk to them and take good notes; you will need them to make accurate predictions.

Analyze trends. Analyze existing trends in your company to see how they have reacted to changes in the past. Try to answer such questions as these:

- What is the company's projected growth rate for the next 3 years?
- What was the company's projected growth rate 3 years ago, and how accurate was it?
- What departments are expected to experience the most growth?
- How much has the company grown in the last fiscal year?
- How does the company plan to achieve projected growth?

Once you have a plan for the future that has a good chance of being accurate, you will have a good project plan in place.

The Bottom Line

Analyze the current business model.

Master It When you are analyzing the current business model, what are some of the types of business processes you should look for?

Determine new business requirements.

Master It A scope document is one of the most important documents you will come up with during your project. What is it used for?

Plan for future business requirements.

Master It What are some of the important factors to consider when planning for future business requirements?

Chapter 3

Planning and Developing Role-Based Security

Protecting information — guarding access to an organization's data — is much like protecting a physical structure. For example, imagine that you own a business and the building that houses it. You don't want the general public to gain access to your building — only your employees should have access. Further, you do not want your employees to have access to every resource in the office once inside. Only accountants should have access to the accounting department, and almost no one should have access to your office. So you put in place various security measures, like alarm systems and locks on the doors.

This concept holds true in Reporting Services. You do not want people to gain access to Reporting Services (your building) unless they have been granted access, and once they are inside you need to make certain that they are restricted from prying in sensitive areas that are not their concern.

In Chapter 16 we will discuss how to give users access to Reporting Services (give them a key to get in the building). In this chapter we will discuss the security mechanisms available to you for securing the resources that users have access to once they have connected to Reporting Services. In our building analogy, this is akin to showing you how to lock the file cabinets.

The ''lock'' that you will learn to use to secure your reports is role-based authorization coupled with Windows Authentication. When combined, these two components allow you to determine who has access to specific resources on your system.

In this chapter, we will assume that you are a member of the local Administrators on the Report Server so that you have appropriate permissions to perform all the tasks described here.

In this chapter you will learn to:

- Plan and develop role-based security as part of your underlying Reporting Services infrastructure
- Understand the role-based security model used in Reporting Services
- Create and modify role definitions
- Create and modify role assignments

Planning Role-Based Security

Think about our business analogy for a moment. Suppose that you are having some contractors build a new building to house your business because it has grown extensively over the past several months. Now suppose that the contractors wanted to hurry up and get the job done so they could

move on to the next project, so they neglected to do the necessary planning. You could easily end up with a building where doors do not fit in their openings, locks are missing from the doors, light switches for one room are in an adjacent room, and so forth. The project would be a disaster.

The same thing can happen to your reporting infrastructure security system if you do not plan well. If you do not configure the security settings properly, then people who need access to a report may not have access, while others may have access to sensitive reports when they should not. This would certainly turn your project into a disaster.

To avoid that scenario, we will show you how to plan your security settings before you implement them. This will help you to implement an effective role-based security system that allows people access only to what they absolutely need.

Gathering Security Requirements

In our business analogy, if you own a business you must ensure that your employees have access to the resources they need to do their work. You should also make certain that your employees do not have access to sensitive resources. For instance, members of your accounting department will need access to the checkbook to pay bills, but members of the maintenance department do not need access to the checkbook for any reason. So you need to figure out who needs access to what, and then grant that access.

The same is true of Reporting Services. Not everyone in your company should have access to every available report. Although some reports may be accessible to everyone, others are sensitive and should be blocked from most users. You need to find out who should have access to which reports and then set appropriate security. How do you find out who gets access to what?

The first place to look for this information is in the business requirements documents that you created at the start of your project, which we discussed in Chapter 2. While you are talking to the users and managers in the various departments, ask them who uses their reports on a regular basis, and who might need to use them later on. This will help you determine the current and future security needs.

Document this information carefully because you will use it to design the security policy for your SQL Server 2005 server later on.

You will also have to look to external regulatory bodies for possible security settings. For instance, if you need to report on credit card information that is stored in your database, then contact the credit card companies to learn the regulations related to generating these reports and who should have access to them. The regulations may require you to allow only two or three people access to the reports and then only to a subset of the data stored in the database. The key is to check with any external regulatory bodies before you start setting security on your reports.

You may be subject to regulations from a number of regulatory bodies at the same time, so be sure to get information from all of them.

Finally, document the existing security settings on the SQL Server (and Reporting Server if you have one). That way, you can be sure that all of the users who have access to databases before you implement your new security settings still have access to those databases after you implement the new settings.

With all of this documentation at hand, you will be ready to implement the new security settings. First, though, let's look at how the security scope was changed in SQL Server 2005.

Setting Security Scope

In SQL Server 2005, security can be applied at one of four different levels, or *scopes*: the server scope, the database scope, the service scope, and the schema scope. If you are accustomed to using SQL Server 2000, you are probably wondering what the schema scope is.

A schema is a new concept in SQL Server 2005. A *schema* is a collection of objects that form a single namespace in a database. Think of a schema as being like a family. A family consists of a number of people, and you know who is in the family because they usually have the same family name. In a schema, you can tell which objects belong to a schema because they will all have the same schema name.

Why do we need a schema name? One short term explains it: name collision. Think about the family analogy again. How many people do you know with the same first name? You probably know more than one person named Bob or Lee or Maria. You would have a hard time talking to them if they were in the same room at the same time if they did not have a family name. In a database, if you have two objects with the same name, you would not be able to address either of them if they did not have a schema name to differentiate them, just like your friends have a family name to differentiate them from other people with the same first name.

This user-schema separation allows SQL Server administrators to perform common administration tasks, such as deleting users, much more easily than they could on older versions of SQL Server. It also means that you will need to take the schemas as well as the users into account in your security plan.

Analyzing Your Security Requirements

You do not want to make changes to the security settings on your SQL Server without knowing how those changes will affect the users. So, before you can implement your security plan, ascertain how it will affect the security settings already in place for your SQL Server 2005 implementation.

The first step in examining security requirements is to determine the scope of each requirement. A requirement should fall into one of the scopes described in Table 3.1.

TABLE 3.1: Security Requirements Scope Criteria

SCOPE	CRITERIA
Server	Anything that references the login to the SQL Server instance or involves the configuration of the instance. Authentication of an individual or service is addressed at this level.
Database	Requirements that address the storage of data in a database, encryption of data, or the security of all schemas contained in a database.
Schema	Application-specific requirements that deal with access to specific SQL Server objects (tables, views, stored procedures, and so on) that will be stored in the same schema and accessed separately from other data in another schema.
Service	Requirements that address the security of a service, HTTP endpoint, or Service Broker queue.

Security requirements vary from company to company and even from department to department, so we cannot give you a comprehensive list of what requirements fall into which scope. Table 3.2 gives a few examples of how some requirements might be classified.

TABLE 3.2: Sample Security Requirements Classification

REQUIREMENT	CLASSIFICATION
Login security must be integrated with Active Directory.	Server level
It must be possible to deny a particular login access to the server if necessary.	Server level
Developers must have read-only access to production database systems.	Database level or schema level, depending on the design of the database
Web services used for reporting to clients must only have access to the Invoice portion of the Sales database.	Schema level if these tables are separated from others by schema; otherwise, database level
Service accounts must be unique for each instance/service combination.	Service level
No user should own any tables.	Schema level
Developers should be able to manage all objects in their development databases.	Database level

All of the items in your requirements documentation should be addressed at some point in your implementation process. For example, any items that fall in the database or schema scopes should be addressed when any database design changes are made.

Dealing with Conflicting Requirements

Many smaller companies do not have rigid security requirements in place; in fact, they may have no security requirements at all. These companies are easy enough to work with when it comes to security because you are creating the security guidelines as you go.

Larger companies usually have security guidelines in place that you will need to follow. The challenge for you is that the security guidelines may differ between departments, regions, or subdivisions. They may even conflict with one another.

You may also have conflicting requirements between your company and external regulating bodies. Suppose that your auditors need to see inventory control reports to make sure that the invoices match what is in inventory. If your company has a policy in place that prevents external users from viewing inventory management reports, you have a conflict that needs attention.

The best place to start when trying to resolve a conflict is at the top. Go to the management or executive staff and find out what they want to do. Of course, you can't just go in with only your word to back you up; take the security requirements documentation and project plan documentation with you. Use this to prove to management that there is indeed a conflict and that they will need to make a decision to resolve it.

If you are asked to resolve these types of conflicts on your own, make sure that you understand the implications of applying one set of rules over another. For instance, if you decide to follow company policy and not allow the auditing firm access to the inventory management reports,

you may have inaccurate audits, which in turn can lead to inaccurate government filings — which can cause serious problems for the company.

When in doubt over what to do, seek out the company's legal counsel. They should be able to offer advice on which set of rules to accept when the conflict is with an external entity.

Analyzing the Costs, Benefits, and Risks of Requirements

You may be wondering why you need to analyze the costs, benefits, and risks of security requirements. From a technical standpoint, it does seem like a waste of time, but from a business standpoint, such analysis is invaluable. You should analyze the costs of security requirements because, monetary or not, all of the requirements have a cost.

Costs

Let's consider a scenario. Suppose that you have a report that your internal salespeople can use to see what is in inventory and available for sale. This report makes it much easier to make sales because the salespeople know what is available. Now suppose that one of your major clients, someone who resells your product, wants to have access to that same report. Giving your client access to that report may at first glance appear to be a good idea, but you should weigh the costs before you grant that access. If you decide not to give the client access to see your inventory, they may decide to work with one of your competitors, one that will allow them access to the information they want. On the other hand, if you do give them access they may look at the report and decide to order an extra 1,000 products because you happen to have them on hand.

That example is not all that complex, and it is easy to see what the costs might be. If you give the customer access you might gain some sales; if you do not give them access, you may lose a client. That translates into monetary cost, but not every cost is monetary.

For example, suppose you have a report that shows where each employee sits in the company and what equipment has been assigned to them. Your maintenance department may use this report to find an employee when they report a problem with a piece of equipment. Some of the other employees may find out about this report and request access to it so that they can find where other employees sit, which would make it easier to find them for meetings. The cost of granting or denying access to this report is not monetary. If you do not grant access to the report, people will have to keep looking for fellow employees the old-fashioned way, which is not as productive as having a report telling them where to look. If you do grant access to the report, other employees may see that some of their colleagues have nicer equipment assigned to them and they may start to complain. Either way, it's a productivity cost, not a monetary one — well, not directly monetary.

Every requirement in your project plan can have a monetary cost assigned to it; you just need to decide what that cost may be. If the cost is in lost productivity, you may decide to average out the hourly pay of those involved and determine how many hours would be lost in productivity. That will give you a monetary cost for productivity.

Take into account other intangible costs as well. For instance, you may lose customers if you do not give them the information they want or if it is too much of a hassle to obtain. You may lose sales if your internal people cannot get to the reports they need to make the sale. If reports just take too long to obtain because of security requirements, you will need to consider the cost of lost work time. So talk to everyone involved so that you include all the possible costs of your security plan.

Benefits

Include a benefits analysis component in your cost analysis report to help show why the security design is necessary. Remember that every piece of security you implement has a

benefit that you can tout. As an example, using roles instead of individual user accounts saves time in administration.

This analysis is important from a business standpoint because security is often seen as a necessary evil — something that just takes too much time to implement and work with. By including these benefits in your security plan, you will help management see the need for a security plan and you will temper these feelings right at the start.

Risk Factors

It is extremely important for you to include a consideration of the risk factors involved in your report because these risks can cost the company dearly if they come to pass.

For instance, suppose that you have an e-commerce website that your customers can use to buy your products. To make purchasing your products easier, your company decides to let the customers save their credit card information on the website for future access. Customers are also allowed to store their addresses and phone numbers on the site for ordering purposes.

Now, suppose that your sales department wants you to generate a report for them that shows the total sales for the day. To do this, you may need to give them access to the database that stores customer credit card and address information. If the security system is flawed, someone in the sales department may get access to the credit card information and sell it or use it for themselves.

This risk has some very high consequences because of the government regulations in place. If your security is breached and someone steals credit card information, in most states you need to report that theft to the government and three major news outlets. You would also have to try to contact your customers directly if possible and inform them of the situation. The government may decide to fine the company for not having proper security in place, and you are sure to lose customers and sales.

Your company may have already taken steps to mitigate such risks by obtaining insurance. The insurance company will want to know what the risk is that they are insuring and what is the likelihood of the risk occurring.

Implementing Role-Based Security

Let's return to our business analogy for a moment. If you own a business, you need to make sure that only the right employees have access to company resources. To ensure the right access, place all of your employees into groups: accounting, sales, maintenance, IT, and so on. Once you have grouped your employees together, you can start granting them access to various resources. For example, accounting can have access to the checkbook, IT can have access to the server room, maintenance can have access to the tool room, and so forth.

Reporting Services role-based security works in the same fashion. You place people with similar needs into groups and then grant access to those groups by assigning them to roles.

> You can also assign individual user accounts to roles, but you will only do that when you want to specifically deny access to a user.

Before you start assigning groups to roles, you should understand exactly what you are securing from a technical standpoint.

Understanding Securable Items

A *securable item* is any item stored and managed by a Report Server that can be secured independently of other items. Other constructs, such as schedules and subscriptions, are not explicitly secured, but are protected within the security of a report. You can create role assignments that will determine who has access to these securable items. Table 3.3 lists these securable items and describes their characteristics.

TABLE 3.3: Securable Items and Their Characteristics

ITEM	CHARACTERISTICS
Folders	Folder security applies to the folder itself and everything contained therein. The Home folder is the root node of the folder hierarchy so the security settings you apply to the Home folder will be initially applied to all subfolders and items. My Reports is a special-purpose folder that is secured through an implied role assignment based on a dedicated role.
Reports	Reports and linked reports can be secured to restrict the actions that a user can take on a report, such as changing the properties of the report. Report history is secured through the report that contains the history. You cannot secure individual snapshots within the report history.
Report models	The role assignment can be specified for all or part of a report model. Typically you will want to secure all model items that map to confidential data.
Resources	Stand-alone resources can be secured to restrict access to the resource itself and its associated properties. Resources that are embedded directly into reports cannot be secured separately from the report.
Shared data sources	These are secured to limit access to the item and its property pages.

Understanding Tasks and Permissions

Consider the business analogy again. As the owner of the business, you assign tasks to your employees. You may assign the accounting department the task of paying the other employees. This task involves a number of other smaller actions, such as writing payroll checks, depositing monies to cover the checks, deducting taxes, assessing vacation time and sick leave, and so on. You need to make sure that your accounting department has all of the necessary permissions to perform the task of paying the other employees.

Tasks are similar in Reporting Services. Each task is made up of a set of predefined permissions that provide access to specific Report Server functions. If a user does not have permission for a function, then they are not allowed to use it. For example, if a user does not have permission to read a subscription for a report, the subscription option is not even available to them in Report Manager.

There are a total of 25 tasks that your users can perform, such as View Reports, Manage Reports, and Manage Data Sources. All of them are predefined, which means that you cannot create your own tasks or modify existing ones.

Figure 3.1 shows how permissions are combined into tasks, and how tasks are then combined into roles that can be used for specific role assignments.

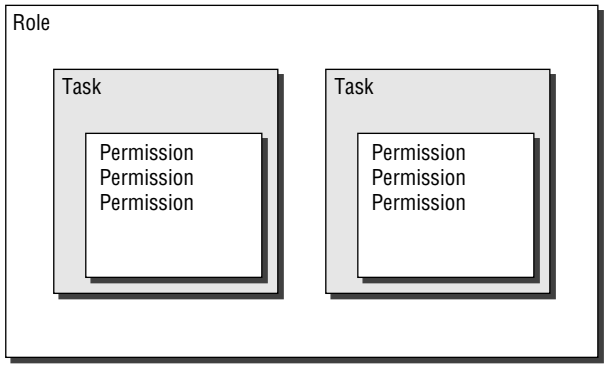


FIGURE 3.1
Permissions and task diagram

There are two categories of tasks: *item-level tasks* and *system-level tasks*. A role can include tasks only from a single category.

ITEM-LEVEL TASKS

These are tasks that are performed on items managed directly by the Report Server. These items include:

- Report
- Folder
- Report Model
- Resource
- Shared Data Source

Item-level tasks are scoped to the Report Server folder namespace. All items that you access through the folders on a Report Server or through URL access are secured by role assignments that include item-level tasks.

Table 3.4 lists item-level tasks, the items to which the permissions apply, and the permissions included in each task. You cannot change these permissions; they are listed here only for informational purposes.

SYSTEM-LEVEL TASKS

System-level tasks are actions that are performed at the system level, such as managing jobs and shared schedules. System-level tasks are scoped outside of the report server folder namespace. Table 3.5 shows the system-level tasks and the permissions included in each task.

In the Reporting Services role-based security model, you grant users permissions through a task. For tasks to be useful, they must be assigned to a role definition.

TABLE 3.4: Item-Level Tasks and Permissions

TASK	SECURABLE ITEM	PERMISSIONS
Consume Reports	Reports	Read Content Read Report Definitions Read Properties
Create Linked Reports	Reports	Create Link Read Properties
Manage All Subscriptions	Reports	Read Properties Read Any Subscription Create Any Subscription Delete Any Subscription Update Any Subscription
Manage Data Sources	Folders Data Sources	Create Data Source Update Properties Delete Update Content Read Properties
Manage Folders	Folders	Create Folder Delete Update Properties Read Properties
Manage Individual Subscriptions	Reports	Read Properties Create Subscription Delete Subscription Read Subscription Update Subscription
Manage Models	Folders Models	Create Model Read Properties Read Content Delete Update Content Read Data Sources Update Data Sources Read Model Item Authorization Policies Update Model Item Authorization Policies Delete Update Properties
Manage Report History	Reports	Read Properties Create Report History Create Report History Delete Report History Execute Read Policy Update Policy List Report History

(CONTINUED)

TABLE 3.4: Item-Level Tasks and Permissions *(CONTINUED)*

TASK	SECURABLE ITEM	PERMISSIONS
Manage Reports	Folders	Create Report
	Reports	Read Properties
		Delete Update Properties
		Update Parameters
	Reports	Read Data Sources
		Update Data Sources
		Read Report Definition
		Update Report Definition
		Execute Read Policy
		Update Policy
Manage Resources	Folders	Create Resource
	Resources	Update Properties
		Delete Update Content
		Read Properties
Set Security for Individual Items	Reports, Resources, Data Sources, Folders	Read Security Policies Update Security Policies
View Data Sources	Data sources	Read Content
		Read Properties
View Folders	Folders	Read Properties
		Execute And View
		List Report History
View Models	Report models	Read Properties
		Read Content
		Read Data Sources
View Reports	Reports	Read Content
		Read Properties
View Resources	Resources	Read Content
		Read Properties

Understanding Role Definitions

In Reporting Services, a *task* is an operation that can be performed on a Report Server. A *role definition* is a named collection of tasks that define the actions a user is allowed to perform on the server.

If you consider the business analogy again, you may ask your accounting department to take on the tasks involved in paying the other employees. In this case, the accounting department is the role definition and paying the other employees is a task assigned to the role.

TABLE 3.5: System-Level Tasks and Permissions

TASK	PERMISSIONS
Execute Report Definitions	Execute Report Definitions (the permission and task name are the same)
Generate Events	Generate Events
Manage Jobs	Read System Properties Update System Properties
Manage Report Server Properties	Read System Properties Update System Properties
Manage Roles	Create Roles Delete Roles Read Role Properties Update Role Properties
Manage Shared Schedules	Create Schedules
Manage Report Server Security	Read System Security Policies Update System Security Policies
View Report Server Properties	Read System Properties
View Shared Schedules	Read Schedules

There are two types of tasks that can be included in a role definition: item-level and system-level tasks. Item-level tasks define what a user can do with an item like a report or a resource. System-level tasks define what actions a user can take at the server level, such as enabling My Reports. A role definition cannot contain tasks from both categories of tasks.

A number of predefined role definitions are available for you to work with. If none of them meet your needs, you can create your own role definitions. First, let's look at the predefined roles that come with Reporting Services.

PREDEFINED ROLES

There are seven predefined roles that allow users to work with the Report Server and the reports stored therein. Five of these roles are for users and two of the roles are designed for administrators. The five roles that are for standard users and their associated permissions are as follows:

Browser Role This is the most restrictive of all of the predefined roles. Users assigned to the Browser role can navigate through the folder hierarchy and open reports, but they cannot create folders or upload new reports.

You can modify the Browser role to better suit your needs. For example, you can remove the View Resources task if you do not want users to see external resources. At a minimum, it is

recommended that this role support the View Reports and View Folders tasks. By default, the Browser role includes the following tasks:

- View Reports
- View Resources
- View Folders
- View Models
- Manage Individual Subscriptions

Report Builder Role The Report Builder role is similar to the Browser role, with one key difference. Users assigned to the Report Builder role have permission to load report definitions from the Report Server into a local instance of Report Builder. This permission allows users to use the Report Builder tool to create ad hoc reports based on published report models.

Like the Browser role, the Report Builder role can be modified to suit your needs. You need to include the View Reports, View Folders, and Consume Reports tasks to maintain the minimum functionality of the role. The default tasks for the Report Builder role are:

- Consume Reports
- View Reports
- View Resources
- View Folders
- View Models
- Manage Individual Subscriptions

To create and modify reports in Report Builder, a user must also have a system role assignment that includes the Execute Report Definitions task.

Publisher Role Users assigned to the Publisher role have the ability to create new folders in the folder hierarchy and upload reports to the server. Publishers cannot change security settings or manage subscriptions and report history. The role is best suited for developers who design reports in Report Designer and publish them to a report server.

As with previous roles, the Publisher role can be modified. You will have to leave the Manage Reports task in place for all members of this role. If your users need to manage folders, data sources, or external resources, you leave the corresponding tasks in place as well. The Publisher Role includes the following tasks by default:

- Create Linked Reports
- Manage Data Sources

- Manage Folders
- Manage Reports
- Manage Models
- Manage Resources

My Reports Role My Reports is a personal workspace that users can use to store and manage reports that they own. The My Reports Role is intended to give users administrative permissions on their own My Reports folder.

It is possible to modify this role as well. You should leave the Manage Reports and Manage Folders tasks in place as well as each of the view-based tasks for minimum functionality. The My Reports role includes the following tasks:

- Create Linked Reports
- Manage Folders
- Manage Data Sources
- Manage Individual Subscriptions
- Manage Reports
- Manage Resources
- View Reports
- View Data Sources
- View Resources
- View Folders

System User Role Members of this role can view basic information about the Report Server, such as schedule information in a shared schedule. They can also execute report definitions in Report Builder. These users do not have the ability to make modifications to the system.

This role can be modified to better suit your purposes as well. The only task you should remove, though, is the Execute Report Definitions task. This will prevent users from using Report Builder to execute report definitions, but they will still be able to view system information. The System User Role includes the following tasks:

- Execute Report Definitions
- View Report Server Properties
- View Shared Schedules

Predefined Management Roles

In addition to the five predesigned roles designed for standard users, there are two roles designed for administrators: the Content Manager and System Administrator roles.

By default, a user who is a local system administrator is automatically assigned to the Content Manager and System Administrator roles. That person can assign additional users to either role to delegate management responsibilities.

Content Manager Role The Content Manager role is intended for users who need to manage reports and web content but do not need to author reports. Members assigned to this role can deploy reports, manage report models and data sources, and decide how reports are used. While Content Managers can perform any item-level task, they cannot perform system-level tasks. The Content Manager role includes the following tasks:

- Consume Reports
- Create Linked Reports
- Manage All Subscriptions
- Manage Data Sources
- Manage Folders
- Manage Individual Subscriptions
- Manage Models
- Manage Report History
- Manage Reports
- Manage Resources
- Set Security Policies for Items
- View Data Sources
- View Folders
- View Models
- View Reports
- View Resources

You can modify the Content Manager role, but Microsoft recommends that you do not.

System Administrator Role Users assigned to the System Administrator role have permission to perform system-level tasks, but they cannot work with server content. The System Administrator role comprises these tasks:

- Execute Report Definitions
- Manage Jobs (e.g., viewing and canceling running jobs)
- Manage Report Server Properties (such as enabling My Reports or setting report history defaults)

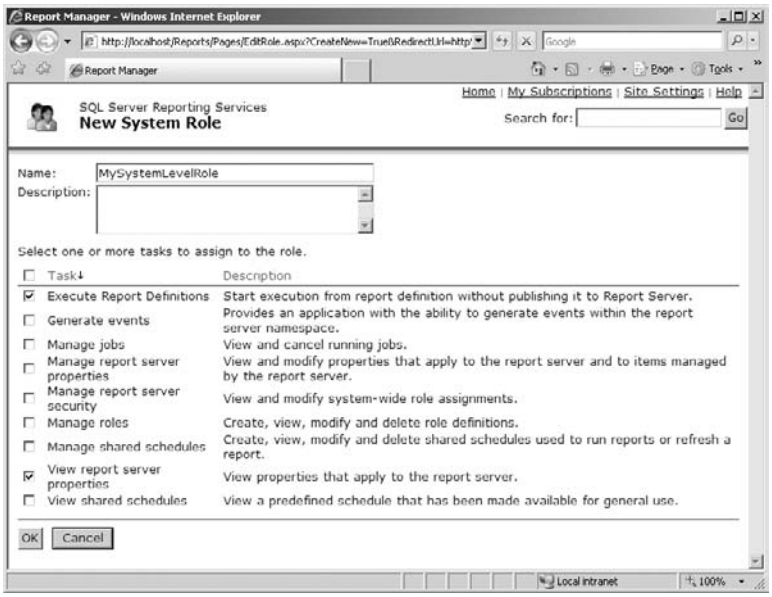
- Manage Roles
- Manage Shared Schedules
- Manage Report Server Security

Customizing Role Definitions

If the predefined roles provided with Reporting Services do not meet your needs, you can modify them. You also have the ability to create entirely new role definitions and delete existing ones. In the next series of steps you will see how to create a system-level role definition using Report Manager.

Creating a System-Level Role Definition with Report Manager

1. Open Internet Explorer and browse to `http://localhost/Reports/` to open Report Manager.
2. Click Site Settings.
3. Select Configure System-Level Role Definitions.
4. Click New Role.
5. Type **MySystemLevelRole** in the Name box.
6. Check the boxes next to Execute Report Definitions and View Report Server Properties to add the tasks to the role definition, as shown here:



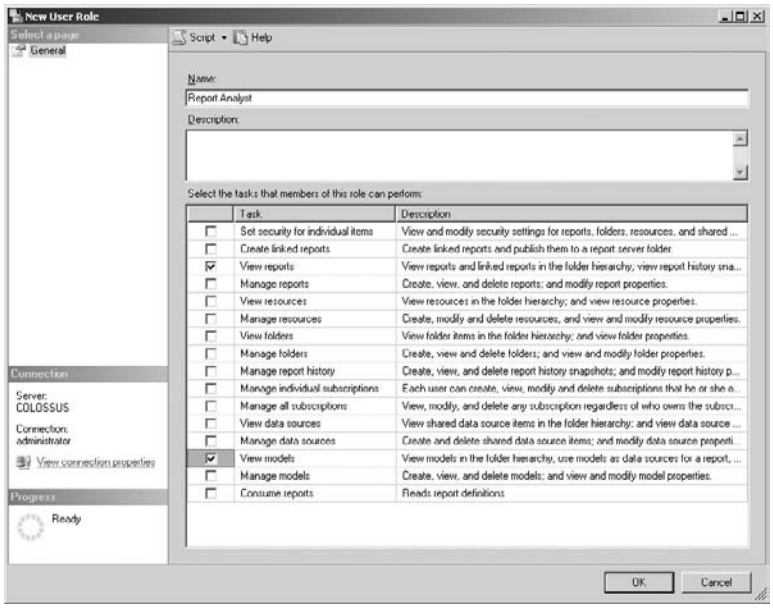
7. Click OK to save the role definition to the Report Server.

Modifying a System-Level Role Definition with Report Manager

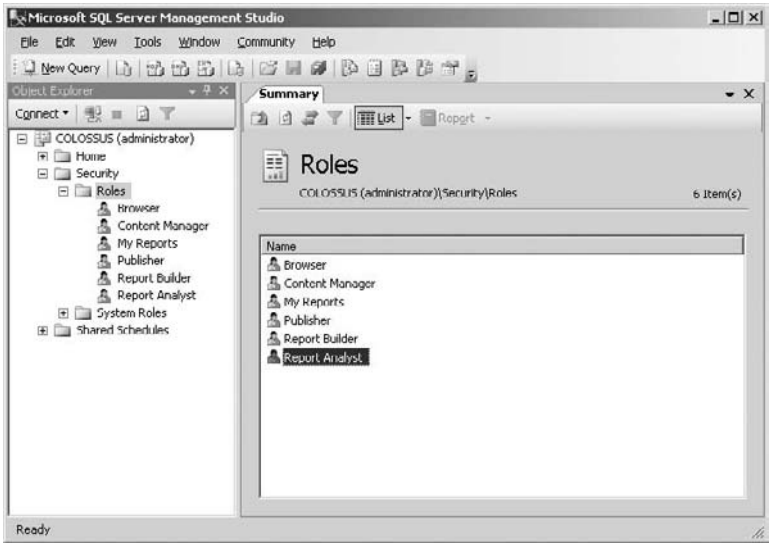
1. Open Report Manager and click Site Settings.
2. Select Configure System-Level Role Definitions.
3. Click the MySystemLevelRole role definition you just created.
4. Modify the task list or the description. Notice that you cannot change the name.
5. Click OK to save the changes in the role definition to the Report Server.

Creating an Item-Level Role Definition with Management Studio

1. Open Management Studio and expand the Report Server in Object Explorer.
2. Expand the Security folder.
3. Right-click the Roles folder and select New Role to open the New User Role properties sheet.
4. In the Name text box, type **Report Analyst**.
5. In the task pane, mark the checkbox next to the View Reports and View Models tasks to add them to the role definition:



6. Click OK to create the role:



Modifying an Item-Level Role Definition with Management Studio

1. Open Management Studio and expand the Report Server in Object Explorer.
2. Expand the Security folder.
3. Right-click the Roles folder and select the role definition for Report Analyst.
4. Open the User Role Properties by double-clicking Report Analyst or right-clicking it and selecting Properties from the menu.
5. In the task pane, mark or clear the checkboxes next to the tasks you wish to change for this role. You can also modify the description but you cannot change the name.
6. Click OK. The updated role definition is saved to the Roles folder.

You can delete a role definition by navigating to it, right-clicking it, and clicking Delete.

Keep It Simple

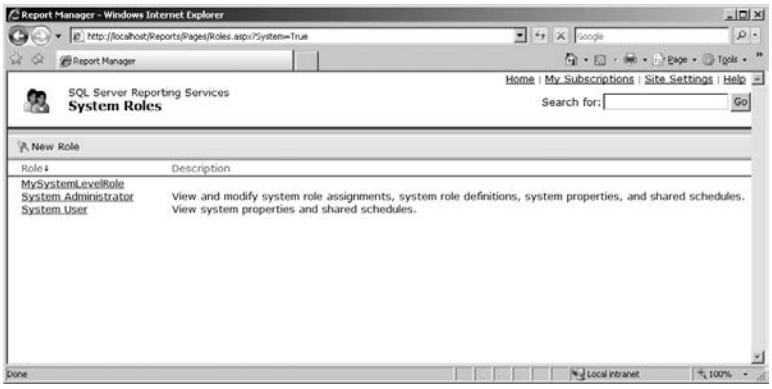
Remember, just because you *can* create new role definitions, that does not mean you *should* create them. Make certain that you really need a new role definition first.

If you create a new role definition every time your security needs change, serious problems will result. First, you will end up with duplicate role definitions. This is a problem because you will not be sure which role to use for a given scenario, and that makes the system harder to manage. Also, you may start canceling out other role definitions through multiple memberships.

In other words, you may add a person to two different roles and end up giving them too many permissions.

Here's the bottom line: be careful. Do not add or modify roles just because you can. See if you can use what you already have in place before creating new role definitions.

You can view all of the existing role definitions on your system in Report Manager. As shown here, just open the Item-Level Roles page or the System-Level Roles page in Report Manager, or open the Security node in Management Studio.



Now that you know how to create and modify role definitions, you should know how to use them. To use role definitions, you must create role assignments.

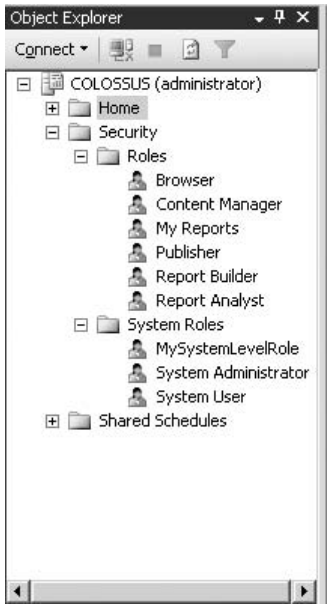
Creating Role Assignments

A role is only useful when it has been assigned using a role assignment. When a user has been assigned to a role, the role controls what the user can and cannot do on the Report Server.

Each role assignment consists of three parts:

- The securable items you want to control access for, including folders, resources, report models, shared data sources, and of course reports

- A user or group account that can be authenticated, usually through Windows security
- A role definition, which describes the set of tasks the user can perform on the securable items



With the exception of the Home folder, all folders, reports, and other securable items inherit their role assignments from the parent folder.

Some role assignments are created for you during the Setup process. This is called default security, and it allows you to access the server after Setup is complete. Specifically, default security maps the built-in Windows administrators group to the System Administrator and Content Manager roles. This is done because the built-in Windows groups cannot be deleted, so Setup knows these groups will be available.

To grant other users access to the server, assign other Windows groups to Reporting Services roles. For example, you could assign the special Domain Users Windows group to the Browser role to grant all authenticated users permission to view reports and folders.

Table 3.6 shows a combination of roles that are easy to define if you want to provide limited access to a large group of users right away.

The built-in Everyone group includes everyone who has access to resources on your network, whether or not they are authenticated.

For most organizations, four role assignments will be enough to handle the security needs of the majority of the user base. You may need more than four if you have folders that require special security settings. For example, you may have a payroll report that only Human Resources personnel are allowed to view. In that case you will have to create a special group in Windows for Human Resources and a special role in Reporting Services for the securable items, and then assign the Human Resources Windows group to the Human Resources Reporting Services role.

TABLE 3.6: Basic Role Assignments

BUILT-IN GROUP	ROLE	SECURABLE ITEM
Everyone	System User Role	System
Everyone	Browser Role	Folder hierarchy starting at the Home folder (the root node)

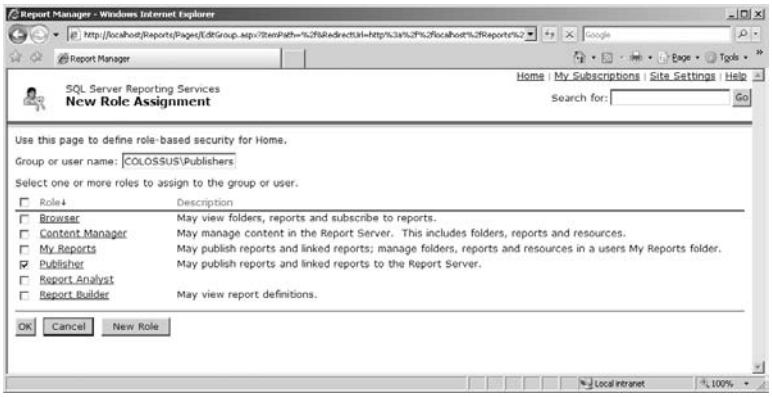
CREATING AND MODIFYING ROLE ASSIGNMENTS

The recommended approach to implementing an effective role-based security design is to start with the default security settings and modify them to suit your needs. It is best to follow the ''set security by exception'' principle. In other words, change or add security to accommodate special cases, but not otherwise.

To define or modify roles and role assignments, use can use either Report Manager or Management Studio. In the next few series of steps you will create and modify role assignments using both methods.

Creating Role Assignments with Report Manager

1. Open Report Manager and navigate to the Home folder.
2. Select the Properties tab to open the Security page for the Home folder.
3. Click New Role Assignment.
4. In the Group or User Name text box, type the name of a valid user or group in the *DomainName**UserName* format.
5. Check the Publisher box under Role:

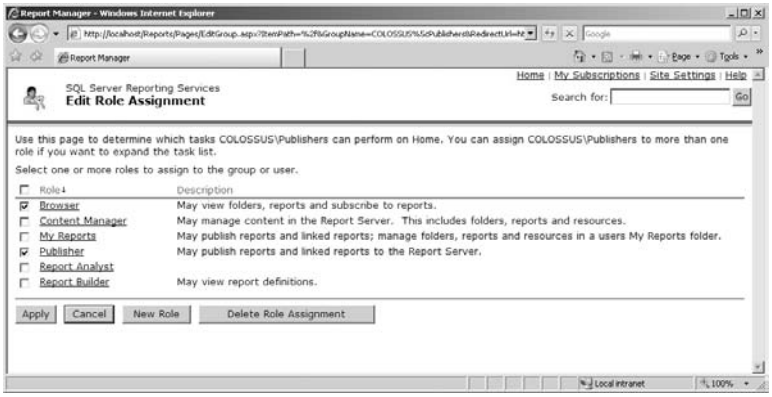


6. Click OK to save the new role assignment and return to the Securities page.

Modifying a Role Assignment with Report Manager

1. Open Report Manager and navigate to the Home folder.
2. Select the Properties tab to open the Security page for the Home folder.

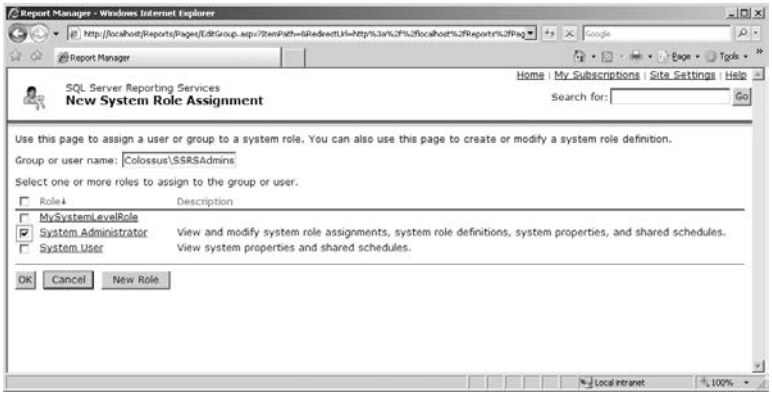
3. Click Edit next to the Publisher role assignment you created earlier.
4. On the Edit Role Assignment page, click the Browser checkbox to add that role:



5. Select Apply to save your changes to the Report Server and return to the Securities. Your new role assignment includes both the Publisher and Browser roles.

Creating a System Role Assignment with Report Manager

1. Open Report Manager and click Site Settings.
2. In the Security section, select Configure Sitewide Security to open the System Role Assignments page. Click New Role Assignment.
3. In the Group or User Name text box, type the name of a valid user or group you want to assign a system role to in the ***DomainName\UserName*** format:



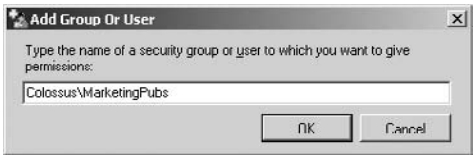
4. Check the box next to the System Administrator task.
5. Click OK. The new system role assignments are saved to the Report Server and you are returned to the System Role Assignments page.

Modifying a System Role Assignment with Report Manager

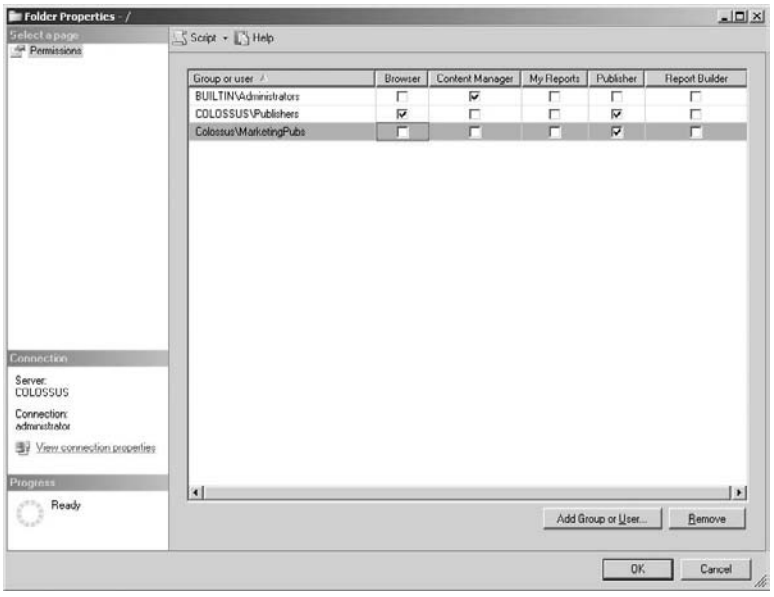
1. Open Report Manager and click Site Settings.
2. In the Security section, select Configure Sitewide Security to open the System Role Assignments page. Click Edit next to the new security role assignment you just created.
3. On the Edit System Role Assignments page, check the box next to the System User task.
4. Click Apply.

Creating a Role Assignment with Management Studio

1. Open Management Studio, connect to your Report Server, and expand the Report Server in Object Explorer.
2. Right-click Home and select Properties to open the properties sheet to the Permissions page.
3. Click the Add Group Or User button to display the Add Group Or User dialog box.
4. Type the name of a valid user or group you want to assign a role to in the ***DomainName\UserName*** format:



5. Click OK.
6. Check the box in the Publisher column to add the task to the role assignment:



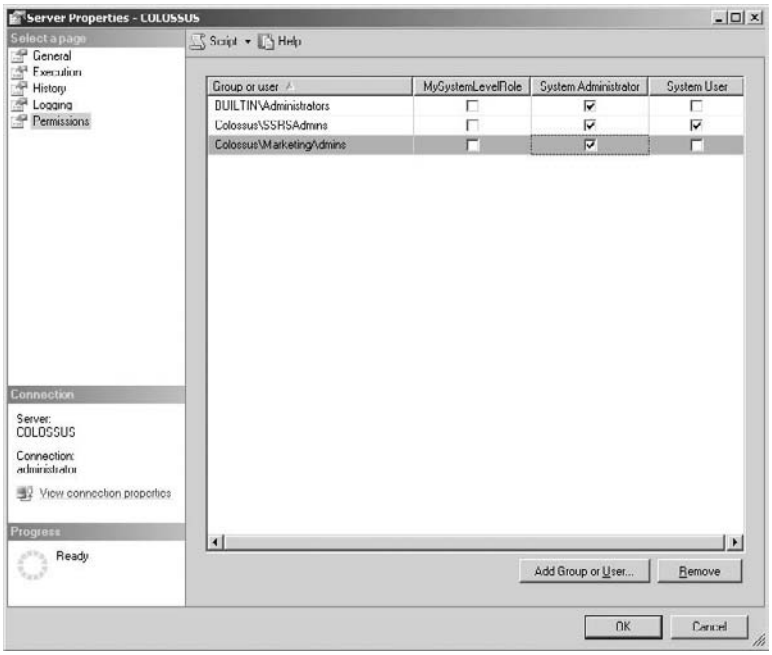
7. Click OK to save the new role assignment to the Report Server.

Modifying a Role Assignment with Management Studio

1. Open Management Studio and expand the Report Server in Object Explorer.
2. Right-click Home and select Properties to open the properties sheet to the Permissions page.
3. Select the role assignment you just created and change the roles you assigned by checking the box in the Browser column.
4. Click OK.
5. Check one or more roles you wish to assign to the user or group. Click OK to save the new role assignment to the Report Server. If you want to delete the highlighted role assignment, click Remove.
6. Click OK to save the modifications to the role assignment to the Report Server.

Creating a System Role Assignment with Management Studio

1. Open Management Studio.
2. Right-click the Report Server and select Properties.
3. Select the Permissions page.
4. Click the Add Group Or User button to display the Add Group Or User dialog box.
5. Type the name of a valid user or group you want to assign a system role in the *DomainName\UserName* format.
6. Check the box in the System Administrator column:



7. Click OK to save the new system role assignment to the Report Server.

Modifying a System Role Assignment with Management Studio

1. Open Management Studio.
2. Right-click the Report Server and select Properties.
3. Select the Permissions page.
4. Select the role assignment you wish to modify and make changes by selecting or deselecting the checkboxes for the relevant roles. If you wish to delete the role assignment, select it and click the Remove button.
5. Click OK to save the modifications to the system role assignment to the Report Server.

Just as you would not want all of the employees in your business to have access to all the resources in your building, you do not want all your users to have access to all your reports. So, just as you would lock your doors and file cabinets in your office building, you can lock the reports and resources in your Report Server. There is just one last area to pay attention to: minimum access permissions.

Exploring Minimum Access Permissions

As you have seen, you have the ability to modify the default role assignments and custom role assignments that you create. However, you cannot remove all role assignments because that would leave your Report Server unsecured, which means anyone could gain access. So Microsoft designed Reporting Services to require minimum security settings.

You are required to have at least one system role assignment that defines access at the system level and one item-level role assignment that defines access to the folder hierarchy.

Also, to prevent a complete, accidental lockout from the system, members of the local Administrators group always have access to the Report Server. Regardless of what role assignments are in place, these administrators can change system settings.

Local administrators have a few other privileges as well:

- Local administrators have read permission on all items stored in the Report Server database.
- Local administrators have read and write permissions on security settings, which means that they always have permission to add, delete, and modify the role assignments that are set at the system and item levels.
- Local administrators are granted elevated permissions permanently. Modifying role assignments does not affect permissions that are granted to members of the local Administrators group.

This does not mean that local administrators have full access to all the reports and data stored on the server, though. To ensure that a local administrator or other user with higher-level permissions cannot access confidential reports, you must secure the reports at the data-access level, requiring users to provide credentials to view the report.

Real World Scenario

CASE STUDY: ALSORANN INDUSTRIES

You have been assigned to develop a new Reporting Services architecture for AlsoRann. To start the project, you schedule meetings and interviews with the management and executive staff to get a better understanding of the security requirements for the new infrastructure.

In the course of your analysis, you find that AlsoRann is a publicly held company, so they need to conform to the Sarbanes-Oxley requirements. They also have a public website on which customers are allowed to store personal information, such as addresses and credit card information, so the company needs to adhere to the requirements of the various credit card companies and regulatory bodies as well.

The Bottom Line

Plan and develop role-based security as part of your underlying Reporting Services infrastructure.

Master It In a similar situation to this case study, consider how you would go about validating your security plan. Who might you consult with to ensure that your plan is valid and meets all the necessary criteria?

Understand the role-based security model used in SQL Server Reporting Services.

Master It Describe tasks, permissions, role definitions, and role assignments.

Create and modify role definitions. A role definition is a named collection of tasks that specify which tasks a user is allowed to perform in connection with a folder, report, or other item. While Reporting Services includes several predefined roles to accommodate various categories of users, you may have to create additional roles to meet your security needs.

Master It Using Management Studio, create a new item-level role definition called Report Reviewer.

Create and modify role assignments. Reporting Services provides default security through role assignments that are configured during setup. These role assignments define access for local administrators. Typically, you will create additional role assignments for folders or reports that relate to specific users or groups; for example, you may want to add role assignments to a group's folders that give individuals in that group the capability to manage those folders and their contents.

Master It Using Report Manager, create a new role assignment.

Chapter 4

Managing User Expectations

Think back to when you were a child, to the first time your family went on a vacation to someplace exciting. Your parents probably told you about the trip a few months in advance. They might have told you about some of the fun things you would do when you got there and maybe even described the interesting things you would see along the way. As time went on, you probably grew pretty excited about the trip. Why? You had started to build expectations about the vacation.

Every time something new comes our way, we start to develop certain expectations — and the bigger the event, the greater our expectations. This is important for you to remember when rolling out a new reporting infrastructure. Because it is new, your users will start to build their own expectations, and because it is a big event they will usually have equally great expectations.

The expectations your users build are powerful, and they can be either high or low. You will need to learn to decipher the kind of expectations your users have and help manage those expectations.

It is usually good for your users to have high expectations, but don't forget: the bigger the event, the higher the expectations. Oftentimes, your users will develop expectations that are too high, which means they may expect too much from your new reporting system. If this happens, your users will not be happy when the finished system does not meet their lofty expectations. Your goal is to help them to keep reasonably high expectations so that they do not get discouraged when the project is finished.

Your users may have been through a similar implementation before and it turned out badly. If this is the case, then your users might develop low expectations. This sounds fine at the outset because you may think that users with low expectations are easily impressed. However, this is almost never the case. Users with low expectations tend to have low opinions of the project from start to finish, and they wind up disappointed with the end result. You will need to help these people to see the value of the new system and help them to build anticipation for the finished product so that they are not disappointed with the end result. Figure 4.1 shows where you want your users to be on the expectation scale.

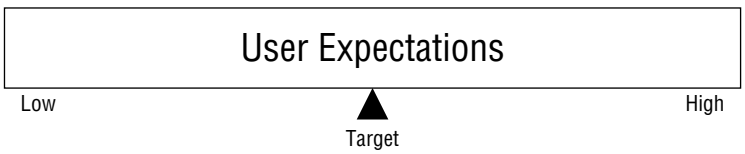


FIGURE 4.1
Try to help your users maintain reasonable expectations.

Before you can help your users manage their expectations, you need to get to know your users better. This does not mean that you have to know them individually; after all, that is usually impossible (especially in large organizations). But you need to get to know them as a group and to develop a feel for what your users in general are saying about the project. It may help to get

to know them as several groups, breaking them up into departments, functional groups, offices, or whatever works best for you. Once you get to know your users better, you can start managing their expectations better.

In this chapter you will learn to:

- Communicate effectively with your users
- Provide education for your users
- Develop and use service-level agreements

Understanding Your Users

As a group, we computer professionals sometimes forget that we are not just working with machines; we also work with the people who use them. This is an important fact to bear in mind as you implement your new reporting infrastructure because the people you work with, your users, have a profound impact on the success of your project.

In fact, if your users are not happy with the finished result, then the project is not a success. Sure, the software may run flawlessly, the security may be bulletproof, and the reports may be works of art. But if the users are not satisfied with the system, they will not use it to its full potential, so it is not truly a success.

To make your project a success, make sure that the majority of your users are satisfied at the end of the project. To accomplish this, you need to get to know them better and try to understand them. How can you do that?

To understand your users better, you first need to think of them as people, not just the computers' operators. This means that you must take into account their thoughts, ideas, emotions, concerns, and all the other things that make them, and you, human.

Like it or not, emotion is a powerful force in the workplace. Consider this: when you are depressed, how productive are you at work? How productive are you when you feel happy? Our emotions affect us no matter where we are, so we need to take them into consideration. That's not to say that you have to become a psychotherapist and listen to your users share their feelings all day and night. It is not feasible to get to know each and every one of your users well enough to gauge their emotional state, and it is way out of the scope of this book to discuss such things. Nevertheless, a broad understanding of how your users may be feeling can certainly help. If they seem to be stressed about something, try to take that into account when working with them. This can go a long way in helping to make them happy with the finished product.

Emotions at Work?

Many people do not believe that emotions have a place in the workplace. Nothing could be further from the truth.

In a paper titled "Poor Judgment in Spite of High Intellect: Neurological Evidence for Emotional Intelligence" by Bechara, Tranel, and Damasio, researchers performed an interesting experiment to show the importance of emotion (*The Handbook of Emotional Intelligence*, September 2000).

In this experiment, subjects were asked to pick cards from decks that were stacked. Some decks were stacked in their favor; some decks were stacked against them.

Some of the subjects in the experiment were specifically chosen because they had damage to the emotional control center of their brain.

Each of the subjects in the experiment, brain damaged or not, was able to figure out which of the decks helped them and which did not. The persons with the brain damage could not stop picking from the bad decks, even though they knew the decks were bad.

What does that tell us? Emotions have an effect on our decision-making process and definitely have an impact at the workplace.

You also need to listen to your users' ideas and thoughts when they share them with you. Many of us tend to think that, because we are the computer professionals, the users should be listening to us. That can be a mistake. You may know the back-end systems by heart, but the users know what they want to see on the front end and they could have some good ideas that you can implement. That is not to say that you should implement every suggestion your users make — just that you should listen to those suggestions and consider their merit.

Communicating with Your Users

The ancient Greeks used to send messengers on foot to deliver messages, some Native Americans used smoke signals, and people in the Old West sent messages via the Pony Express. This shows that communication has always been a vital aspect of life and people were willing to put forth a great deal of effort to make it happen.

That fact has not changed: communication is vital to every aspect of our lives, especially so for complex projects such as implementing a new reporting infrastructure. Communication is so important that, without it, your project will fail. So, you need to be able to communicate effectively with your users, management, staff, and coworkers. Communication can take many forms, including:

- One-on-one communication
- E-mail
- Phone calls
- Group meetings

To effectively manage your project, you will need to hone your communication skills. Let's discuss a few helpful points.

Communicate Early and Often

The most important thing you can do when trying to communicate with your users is to actually do it. Many of us do not like going to meetings when we could be working on the new reporting system, but meetings are a necessary function of any project. So, how many meetings are we talking about having?

The first activity you should plan for your project is a kickoff meeting. This meeting does not have to be very long, but it is an important first step. At the least, you should invite all of the key stakeholders to this meeting. If you have the facilities, you could invite everyone involved in the project. During the meeting you can discuss what people can expect to happen during the life of the project and how they will be affected. You do not need to go into great detail; a high-level overview should be enough to satisfy everyone's curiosity.

If you have more than one geographic location, you may want to schedule a kickoff meeting in each location to make sure no one gets left out, or even feels left out. It may be a good idea to schedule a conference call or a virtual meeting using an online meeting service if you cannot physically attend a meeting in each location.

Once the project gets under way, make certain that there is a constant flow of communication. Varying schools of thought exist on how often to communicate; your circumstances may dictate how often you can meet, but a good rule of thumb is once a week. You may need to meet more often than that, but almost never less often.

Unlike the kickoff meeting, the weekly meetings will ordinarily involve only the key stakeholders in the project. Occasionally you may have to invite some users, but it is best to keep the meetings small so they are easier to manage. When you have your meetings, it's important that they appear valuable to everyone in attendance. You can do this by coming up with an agenda and sticking to it. An agenda may look something like Figure 4.2.

FIGURE 4.2
Use an agenda to keep your meeting on track.

Weekly Status Meeting for 1/10/2007

9:00 Server installation status - Bob Thompson
9:15 Requirements gathering status - Jane Worthington
9:30 New training curriculum status - Mary Chen
9:45 New issues (Q&A) - Joe Jorden

Your coworkers could be aware of issues that have not come to your attention, so it's a good idea to gather input from them about items that should be placed on the agenda. Adding a question-and-answer session to your meeting is another good idea. You do not need to do this for every meeting, but once a month or so it is wise to let everyone ask those questions that you may not have been able to cover yet. When you schedule a question-and-answer session, schedule more time for the meeting to make sure everyone is heard.

It is going to be important for you to stick to the scheduled agenda for your meeting. If your meetings are constantly running long, then the scheduled attendees will stop coming because they know that they will not be able to ''escape'' the meeting to get back to work or make their next appointment.

You also need to make sure that your meetings stay on topic. Occasionally, someone brings up an issue that is not on the agenda and must be discussed right away. However, most of the time people bring up issues that do *not* need to be discussed right then and can therefore be scheduled for discussion at the next meeting. If you don't keep your meeting on track, many people will stop attending because they know that most of the meeting topics will not apply to them anyway.

So how do you keep your meetings on track and on time? Make sure everyone follows the agenda. Before you have the meeting, circulate the agenda so that all the attendees have a copy. That way, everyone is prepared for the meeting in advance. Then, during the meeting, if someone just won't stop talking, you can tactfully remind them that their time is up by saying something like, ''Bob, I'm sorry to interrupt but it's 9:15.'' That simple statement lets Bob know that it is Jane's turn to speak and that he should wrap up his segment. Also, if someone starts a discussion about something that is not on the agenda, you can say something like, ''That's a good point; let's add that to next week's agenda so we can give it the attention it deserves.'' This statement should remind the person that they are veering off-topic and that there is a better time to discuss their concern.

Real World Scenario

CASE STUDY: TALKSALOT COMMUNICATIONS

Talksalot Communications began life as a small company — so small that the entire company met once a week in the conference room for status updates and to discuss outstanding issues.

The weekly meeting started at 9:00 AM. Until 9:15, the owner of the company (who organized the meeting) would explain the latest movie he'd watched and tell the employees why they should watch the movie as well. He would then ask everyone in each department what they were working on, usually eliciting an answer like ''the usual stuff.'' The meeting was scheduled to last an hour but usually went on for 90 minutes.

I mentioned to the owner of the company that the meetings seemed to be disorganized and that nothing of much consequence was being accomplished. I suggested using an agenda to keep the meetings focused and on time.

The owner used an agenda for many of the meetings that followed and everything improved. The meetings stayed on track — no more veering off to discuss movies. Meetings ended on time or early, and everyone who needed to speak got the chance.

Be Honest and Forthright

Here's an interesting tale a colleague of ours once told. A computer was having problems in one of the offices where this consultant was working, so the office personnel there asked him to take a look. He examined the machine and told them, ''I need to reinitialize the operating system from a zero wait state.'' One of the office workers looked him squarely in the eye and asked, ''You mean you're going to reboot it, right?'' That was the last time our colleague used that phrase with anyone in that office.

That anecdote illustrates a good point; we need to be honest and forthright. It is easy for us to think that the users don't understand technical detail, so when we get tired of their questions we may just start throwing a lot of technical jargon at them in hopes that they leave us in peace for a few minutes. Also, we sometimes think that we need to have all of the answers or the users won't respect us. So when they ask a question that we do not have an answer for, we conjure up something that sounds plausible.

Whatever the reason, users can tell if you are not being forthright with them, and they will start to lose respect for you. That attitude is disastrous for your project and for your career. Train yourself to be honest and forthright no matter what the circumstances. If you need to get some work done and you are losing patience, it is better to explain that to the user by saying something like, ''I would love to discuss that with you right now, but I have some other pressing matters I have to attend to; can we talk tomorrow?'' If a particular question has you stumped, tell the user you do not know but you can look into it for them. The person knows you are only human, and they will respect you more when you realize that as well.

Educate Your Users

As we explained earlier, when people know that something new is going to happen to them they start to build expectations based on past experience. You cannot change their past experiences so, in order to manage their expectations, you need to change the way your users formulate those expectations. You must educate your users so that they base their expectations on current facts instead of past experience.

Many of us think we should wait until the system is in place so we can educate the users by using the finished system. By the end of the project, your users have already formed their expectations and they will be reluctant to change them. It is vital, then, that you think of education as an ongoing process. You should start educating your users during the project kickoff meeting, where you should explain what they can expect to happen during and after the project. Then keep that education going during the weekly status meetings.

Because the status meetings are for key stakeholders, you should send a weekly status e-mail update to the user community in general.

There is more to education than just weekly status meetings and project updates. Make sure that your users know what they can expect from the system once it is complete: how the reporting application works, what the finished reports will look like, what functions are available to them, what security settings are in place, and so on. One way to accomplish that is by creating storyboards and prototypes.

Prototyping and Storyboarding

Many people learn new things visually: they have to see the new things in order for the concept to sink in. If they can't visualize the new concept or physically see it, they may never fully understand it. So, to ensure that your users receive the best education during the development process, give them something to look at: a storyboard or prototype.

The VAK Learning Style

- The visual-audio-kinesthetic (VAK) learning style is a neuro-linguistic programming (NLP) model that examines how the human mind processes information. In a nutshell, this model states that all information we take in is processed through our senses. All five senses are used, but we use three primarily for learning:
- Visual
- Audio
- Kinesthetic

According to this model, people who learn visually must be able to see what they are learning, or at least be able to picture it in their mind. Audio learners prefer to hear information. Kinesthetic learners understand best when they can get physically involved in what they are learning.

It should be noted that, while this model is in use in a large number of reputable schools and educational centers, there are studies that question its accuracy and usefulness — a debate that goes beyond the scope of this book. Due to the widespread use and success of this model, though, we thought it was important to mention it.

Storyboards

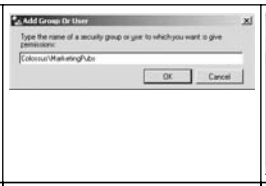
You may be familiar with the concept of storyboards because they are used in animated television and movie productions quite a bit. A storyboard created for a movie production resembles a comic strip in a newspaper; each scene in the movie is sketched out on paper (or computer) so that the director and actors can see what the movie should look like when it is done.

A storyboard used in systems design is similar, except that we are not sketching scenes of a movie but screens of an application. For a Reporting Services implementation project, you can educate your users by creating a storyboard of the finished application. You do not need to draw each screen; you can take successive screenshots using an image-capture application and present them to the users.

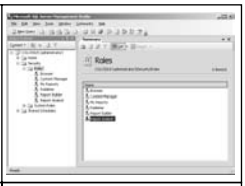
You can use the PrtScrn key on your keyboard to capture a screen image, but it is a good idea to purchase an image-capture application (e.g., Snagit from TechSmith) because they give you more flexibility in what you capture and how you do so.

FIGURE 4.3
Use storyboards to educate your users.

Screen 1: The System Roles page



Screen 2: Add Group Or User



Screen 3: Management Studio

For example, if you are trying to explain to a group of administrators how to assign a user to a role (as we discussed in Chapter 3), you could go through the process yourself on a test machine and take screenshots at every step. You can then print these screenshots and arrange them on a board, or just put them in a Microsoft PowerPoint presentation and explain each screen. You'll be amazed at how much easier it is to educate your users when they have something visual to focus on. Figure 4.3 shows you what a storyboard might look like.

PROTOTYPES

A prototype is an incomplete model of a finished product, which in this case is a report. There are several variants of the prototyping process, but they all stem from two basic types: throwaway and evolutionary.

Throwaway Prototyping This process is called throwaway prototyping because you create a prototype that you have no intention of keeping — you throw it away. This methodology is useful because it allows you to create prototypes quickly. You do not need to worry about getting everything ''just right'' or making sure the data is accurate, because the prototype is just going to be tossed out anyway. This works great for showing and getting feedback about layout, font size, image placement, and other settings that do not require accurate data.

Evolutionary Prototyping This process differs completely from throwaway prototyping because the prototypes are not thrown away. You keep the prototypes and work with them until they are finished, at which point they become the finished product. You can use this method when your users need to see accurate data in the report prototype.

Before you create a report, you talk with the user about what they want the report to do. They may tell you what data they need in the report, how they want the data arranged, what format the fonts should be, where the icons and logos should be placed, and the like. Once you have all that information, you could create a working report and get user input, but if they do not like it you have wasted a lot of time. That is where prototypes come in handy; they help you make sure that you and your users are in sync about what is included in the reports that you generate.

Instead of creating a final version of the new report, you can create a prototype to show the user and get their feedback. This is much faster for you because you can fill the report with dummy data, set the fonts and logos, and show the prototype to the user. Each time you create a new prototype, you get closer to what the user needs until finally you come up with the final design for the report. All the while, you are also educating your user.

Most of the time, you will only create one or two prototypes for a report. Instances where more than two prototypes are required for a report are infrequent.

Each time you create a new prototype of a report, you can help the user understand what Reporting Services is capable of doing. For example, say you've created a prototype of a sales

report that the user likes. However, they have just created a new commercial and they want you to embed the movie file into the report so that people can see the movie while reading the report. You can help them to understand that this is just not feasible and there are better ways to accomplish their goal, perhaps by providing a link to the movie file on a network server. Regardless of the requests, the prototype will save you time and help you educate users at the same time.

Training Classes

Training classes are one of the most effective methods available for educating your users about the new reporting system. You can't just send your users to any Reporting Services class and hope for the best, though; you must have a user education plan in place. To develop a feasible education plan, you require two pieces of information:

- What your users need to know about the system
- What they already know about the system

Know what your users will be doing with the system so that you can determine the best classes for them. Find out what they already know about the system so that you do not send them to a class that won't interest them. That would be a waste of time and resources and would bore your students.

Figuring out what your users need to know about the system is the easy part of developing an education plan. You must communicate with both the user and the management in the user's department to find out what they will be doing with the new system. Once you know what their duties will be, you can decide which classes best meet those needs. The trick is deciding whether to create a custom class or have the user attend a standard class offered at a training facility.

There is no hard-and-fast rule about whether to create a custom class or attend standard classes, but there are some rough guidelines to help you decide. If your company has developed a custom application to access and display reports, you will have to create a custom class to train the users on how to work with it because no standard class will be available. If you are using standard applications to access reports, and you have modified the standard applications, you may need to create a custom class. If you have some complex reports, you may consider creating a custom class for training the users how to read them. If you are not using any custom applications and you do not have complex reports, you may not need custom classes.

The next step is trying to figure out what the users already know so that you do not waste their time and frustrate them by having them take classes that do not interest them or teach them anything. One of the best ways to do that is with a quick assessment test to find out what they already know. An assessment test typically consists of 20 to 40 multiple-choice questions that you have users fill out before assigning them to a class. You can have them do this on paper, but you will get a better response by having the users submit the answers on an intranet site.

Example Assessment Test Questions

The questions you put on the quiz will be different at each company, but here are some examples to get you started:

1. What is the URL to get to the Report Manager interface?
   Answer: `http://servername/Reports`
2. What formats can you use to save reports on your local system?
   Answer: PDF, Excel, Web Archive (MHTML), TIFF, XML, CSV

Many software packages are available that you can use to create a custom test and submit the answers online. This approach is usually more effective than creating a system on your own.

If you decide to create a custom class, you will need to create custom courseware to match. Some have made the mistake of thinking that courseware is just another form of documentation, so they decide to write it themselves. However, keep in mind that key differences exist between custom courseware and documentation that make courseware more challenging to write:

- Courseware is a series of timed modules. Documentation does not need to be timed.
- Courseware generally contains assessment questions with answers; documentation does not.
- You need two editions of courseware: a student edition and an instructor's edition of the courseware with all of the answers and instructor notes. You need only a single edition of documentation.
- You need to write hands-on, timed labs for custom courseware. You do not need to create labs for documentation.

Knowing these requirements, you may find it better to hire a firm with professional instructors who have experience in creating custom courseware if you decide that the standard classes are not sufficient and you need custom classes.

Available Classes for Reporting Services

At the time of this writing, Microsoft is offering these courses on SQL Server Reporting Services:

Number	Name	Format
2944	Updating Your Reporting Skills to Microsoft SQL Server 2005 Reporting Services	E-learning
2797	Designing a Reporting Solution Architecture Using Microsoft SQL Server 2005 Reporting Services	Classroom

Regardless of the format you decide to adopt, you will need to ensure that your students are getting the most out of the class. To do so, you will need to do two things. First, give the students a postclass evaluation test. This can be the same test you gave them before the class, or you can make up another test with similar questions. Be sure that you cover the main points that were covered in the class so that you get an accurate reading of the quality of training they received.

The second thing you should do is have the students fill out a satisfaction survey after the class to find out what they thought. This is a generic survey in which you ask questions about the instructor, the material, the classroom, and so forth. Have the users fill out this survey and submit it once the course is complete. You can then use this data to improve the conditions of the class, change the course material, or cancel the class altogether if necessary.

Listen to Your Users

With all of the meetings and training you'll be having, the advice to listen to your users may seem obvious. But there is more to listening to your users than just talking to them and listening to their ideas on reports — you need their feedback and you want to make them feel as if they've been heard.

This is an important part of keeping the users happy and helping to meet their expectations. Consider the following example. Suppose you go out to eat at a nice restaurant and the food is undercooked when it arrives at your table. You call the waiter and inform him of the problem. The waiter then tells you he will get this fixed right away and whisks your plate off to the kitchen. You feel as if the waiter heard and understood your problem. Now, suppose the waiter brushed off your concerns by saying something like, ''You ordered a medium steak, and that's what I brought. Would you like to order something else?'' That would be frustrating because the waiter is not acknowledging the problem as you see it, so you do not feel as if you are being heard.

The same is true of your users; they want to be heard regardless of what caused the problem. Suppose, for example, that a user asks you to create a report with a logo on the top-right corner. So you do just that; you create a report with a logo at the top-right corner. When the user sees the report, however, they think the logo is too big, so they ask you to make it smaller. If you say, ''OK, I'll make that smaller for you'' and then make a note of the request, that is probably the last you will hear about the size of the logo. If you say, ''You didn't specify a size for the logo, so I just put in what I had,'' then the discussion will undoubtedly continue because the user will not feel as if they are being heard.

Of course, this is a simple example. Problems are usually much bigger than the size of a logo. To satisfy your users and help meet their expectations, you need to gather the users' feedback and make sure they know you have gathered it. There are a few ways you can accomplish this.

If you are talking to a user face to face, in a meeting, or one on one, respond to them verbally when they bring up a problem. Simply saying ''I see, I'll work on that'' can go a long way. Follow up that verbal confirmation by making a note (if you have something to do that with). This shows the user that you are taking their request seriously and they feel as if you are trying to help them.

Another method for gathering feedback is a survey, a series of 5 to 10 questions that users can answer quickly. True or false and yes or no questions are excellent choices because they are easy to answer. If you create a survey with complex, multiple-choice questions, you are not going to get many responses. In fact, even with simple questions you can expect only about 1 percent of your users to respond.

Example Survey Questions

Here are some sample questions that you can put in a survey to get user feedback:

1. Do you find the interface easy to use?
2. Is the corporate logo on the reports too big?
3. How often do use report *X* — on a daily, weekly, or monthly basis?
4. Does report *X* render fast enough?
5. Is it easy for you to find the reports you need?

When you implement a survey, be prepared for some negative attitudes. Not everyone is going to be happy with the system, and that will be reflected in the survey. It is natural to take this as

a personal affront; after all, you put a lot of work into that system. Remember, though, this is business and you want to make the business run as smoothly as possible. Do not take negative comments personally; they are not about you or your work; they are about the system.

Remember, too, your users may have some feedback that you cannot do anything about. Some users may not like the color of the company logo, an attitude over which you have no control. Other users may want access to reports that you cannot give them. Regardless of the feedback, make sure your users know you've received it. If you cannot do anything with their feedback, let them know so that they do not expect results.

If your company intranet is based on Microsoft SharePoint Services or Portal Server, you can use the built-in survey tool to create and publish surveys to your users.

Using Service-Level Agreements

Many IT managers measure user satisfaction by the number of help desk calls they receive. If they do not receive many calls, then the users must be happy with the system. However, this is not the best way to measure user satisfaction because this theory assumes that dissatisfied users will call you to let you know they are unhappy.

What users actually do when they are unhappy varies based on the culture of the company and situation. Some users will turn to outside vendors for help if the company allows that. Departments may send their own personnel to classes to learn how to support your systems. Other users may just grin and bear it if they feel that there is nothing else they can do.

One way to prevent these different, undesirable scenarios from occurring is to create a service-level agreement (SLA) between IT and the user community. This contract will be a tremendous help in managing your users' expectations of the system because it sets the bounds of the system. Some of the key concepts defined in an SLA are:

- The availability of the service to the user
- A performance target for the systems
- The bounds of guaranteed performance and availability
- Measurement and reporting mechanisms

This document is important because it helps you not just to manage user expectations but also to set their expectations. Your users will feel better knowing when to expect the system to be available and how quickly you will respond to a problem. It is easy to think of an SLA as a way of stifling complaints, but do not fall into that trap. Think of an SLA as:

- A communications tool
- A conflict-prevention tool

An SLA can help you open communications with your users and keep them open. It can actually ensure that your users communicate with you when there is a problem instead of looking for outside help. At the same time, it also ensures that you have to communicate with the users.

This document can help you to resolve conflicts when they arise because the constraints of service are written and agreed to. For example, if your users tell you that system is not as reliable as you promised, then you can refer to the SLA to see if the system meets the agreed-upon availability target.

Preparing the SLA

Service-level agreements take time and effort; you cannot just sit down, start typing, and hope for the best. You need to define several components in order to craft an SLA:

Parties As with any contract, you need to define the parties to the agreement. The first party is going to be some part of the IT department. It may be the entire department in a small company; in a larger company, it may only be a subset of the IT department. The second party is going to be a user community. In small companies this could include the entire company; in larger organizations you may need a separate SLA for each department in the company.

Services Next, you should define the services to be provided to the users. For the purpose of a Reporting Services SLA, this could include access to the Report Server, scheduled reports, access to Report Builder and report models, database availability, and the like.

Volume This is the volume of the demand for service over time. In other words, you need to define when the users will be placing the heaviest demand on the system. For instance, your users may place a heavy demand on the system from 8:00 AM to 6:00 PM and almost no demand during evening hours.

Timeliness This defines how quickly the users can expect something to be completed after it has started. For example, suppose that one of your more complex reports requires access to data sources located in different buildings connected by a wide area network that is under heavy load during the day. To define timeliness for such a report, you could state that this report will take no more than 10 minutes to render during peak hours.

Availability This is the time window during which service is available to the client. No system will be available 100 percent of the time, so you will need to find a reasonable percentage of availability. For example, if your system is available 99.99 percent of the time on a yearly basis, you will have 52 minutes of downtime per year.

Reliability Reliability is the ability of a device to perform its function without failure. For instance, you want your car to run while you are driving it, but you do not expect it to run 24 hours a day. Your car should be available to you most of the time and should be reliable when it is in use.

Compensation This is compensation for failure to keep the agreement as written. If there is no compensation, then the agreement has no real power. If your company has a chargeback system in place, compensation may actually be monetary. If you do not have a chargeback system in place, this could involve a review of the department by upper management to get things back on track.

Measurement Once availability and reliability are agreed upon, you must spell out how the system performance will be measured. Many tools are available for measuring performance and availability, such as the built-in Windows System Monitor. Regardless of the tools used, IT must take the measurements to verify the availability.

Renegotiation Systems change over time, and reliability and availability requirements change with them. Be sure to renegotiate the agreement at least once a year to keep pace.

Once you have all of the needed information and you and your users have reached an agreement on what constitutes an acceptable SLA, you need to put it on paper and sign it.

Writing Documentation for Users

All of the hard work you have put into communicating with your users will be for naught if they do not remember what you told them. They may not be able to take notes, they may miss something in the meetings or in class, or they may just forget where they put their notes. Regardless of the reason, you need to help your users by providing documentation for them.

User documentation comes in a variety of formats. The most common is the help file. To see user documentation in help file format, just open an application and press the F1 key to open the help file. The method for creating this help file differs based on the software package you use to create it. Some software packages will take a series of Microsoft Word documents and combine them into a help file for you, while others will compile a grouping of HTML files into a help file.

The documentation does not need to be in a help file format, though. You can create documentation in any form and place it anywhere that your users can gain access to it.

One of the first things to include in your documentation is a series of how-to articles. These are just a series of steps that your users will need to perform to accomplish a task. It will look something like the exercises included in the chapters of this book. For an example, look at the steps from Chapter 3 for creating a role definition:

Create a System-Level Role Definition with Report Manager

1. Open Internet Explorer and browse to `http://localhost/Reports` to open Report Manager.
2. Click Site Settings.
3. Select Configure System-Level Role Definitions.
4. Click New Role.
5. Type **MySystemLevelRole** in the Name box.
6. Check the boxes next to Execute Report Definitions and View Report Server Properties to add the tasks to the role definition.
7. Click OK to save the role definition to the Report Server.

The documentation should be simply stated so that it is easier to read. Remember, you are not writing this for technically savvy people. Write the documentation in a way that accountants, maintenance personnel, manufacturing personnel, and customer service representatives can understand.

Be certain to include screenshots of everything you do so that your users can see what the end result is supposed to look like. You should get a professional screen-capture application for this so that you can control what you are capturing more effectively.

Once you have created the documentation, whether it is a help file or a text document, put it somewhere the users can access it. The best place to put the documentation is usually on the company intranet. You can place a link to the documentation right on the home page of the intranet so that it is easy for users to find.

If you use Microsoft SharePoint, you can create a document library for the express purpose of storing the documentation.

That being said, you can see that creating documentation is not a trivial matter. You need to do a great deal of research and format the documentation in a way that is easily understood. It may be prudent, then, to consider hiring a professional technical writer or service to craft the documentation for you.

The Bottom Line

Communicate effectively with your users. If you do not communicate effectively with your users, your project is doomed to fail. You need to get input and feedback from the user community to make sure they know how to use the system and ensure that they are satisfied with it.

> **Master It** How often should you communicate with your users and in what settings?

Provide education for your users. Users will not be happy with the system if they do not know how to use it. To make sure they understand the system, you need to provide education.

> **Master It** What are some of the key differences between standard user documentation and custom courseware used in teaching a class?

Develop and use service-level agreements. A service-level agreement (SLA) is a contract between the IT department and the user community that defines the system availability. This is a protection for you and the users.

> **Master It** What components do you need to define in order to craft a service-level agreement?

Chapter 5

Change Management

It's likely that every aspect of your life has been affected by change at some point. This includes your work life. Change plays a major role in the business world, so you need to work with and manage change during your project.

Make no mistake; with your new reporting system you are introducing change into your workplace. Whether your company has been creating all of its reports on pen and paper or on a mainframe system, they've been using some form of legacy reporting system. You are trying to change that, so you must manage the change process in order to make your project a success.

This is going to be a complex task for you because of the way people react to change. Some feel that change is beneficial and that good results will come from it. Others resist change because they are set in their ways and fear an unpleasant outcome.

To illustrate, think of some of the processes we see in nature. Caterpillars become butterflies, which most people think is a positive change. On the other hand, metal changes into rust when it oxidizes, which most people think of as a negative change (especially if the metal is part of their car).

In business, users tend to view new software systems as a negative change. They think that the change will slow them down and make life more difficult. Your challenge is to help them view the change in a positive way. Figure 5.1 should help you visualize this concept.

The change does not stop when the project is complete. Like all software, your reporting system is always evolving and changing, which means you'll manage that ongoing change as well. This will require you to put change-management controls in place to ensure that the system is not updated randomly by overzealous developers or system administrators. Once your users see that the system is under control, they will accept it much more readily.

This chapter will help you:

- Understand the theory of change
- Learn the skills needed to affect change
- Choose a change-management strategy
- Manage ongoing change

Understanding Change-Management Theory

You may not realize this, but you've been managing change your whole life. Now, though, you have to manage change for *other* people — the employees in your company. To do that effectively, you must understand the basics of change-management theory.

Change management stems from a psychology, business, and engineering background, which means that some models of change management are based on individual change processes and others are based on organizational change processes.

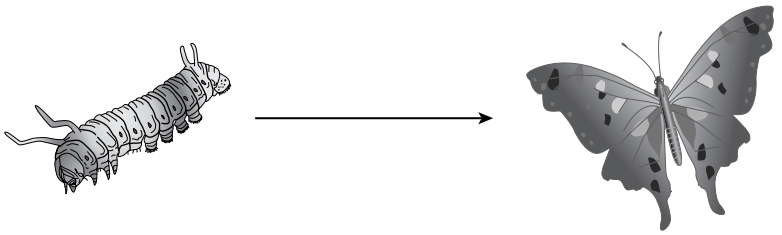


FIGURE 5.1
Help your users to view change in a positive way, like a caterpillar becoming a butterfly.

Unfreezing/Changing/Refreezing

Another frequently used model is the one put forward by noted psychologist Kurt Lewin. In his model, Lewin described three stages of change:

1. Unfreezing, when users recognize the need for change
2. Changing, when changes are put in place
3. Refreezing, when new behaviors are reinforced and solidified

This model applies to your Reporting Services implementation project. Your users are ''frozen'' in the sense that they are accustomed to using the existing reporting system. To ''unfreeze'' them, identify the forces in the users' minds that are resisting change and work to nullify them. For example, they may be longtime users of the current system and they do not want to learn anything new. You can unfreeze this behavior by helping the user see how simple the new system is to learn and use.

Once the users are ''unfrozen,'' it will be easier to make the necessary changes (in this case, implementing Reporting Services). After the changes are in place, and even during the process of implementing the changes, you can start to ''refreeze'' the users.

Do not misunderstand the concept of ''refreezing.'' This is not a permanent state of mind that you cannot undo. Refreezing means that you will reinforce the new behaviors while still being open to feedback and preparing for future change. For your project, that involves getting the users to work with the new system regularly and communicating with them frequently about updates and improvements. Figure 5.2 illustrates this concept.

ADKAR

Organizational change management takes individual change management to the next level. With organizational change management you are now managing change for an entire organization instead of just one person. There are a number of organizational change management models. One such example is Prosci's ADKAR model, which includes five building blocks (as shown in Figure 5.3):

- Awareness — why change is necessary
- Desire — to support and be involved in the change
- Knowledge — how to make the change
- Ability — to implement the new skills and behaviors required for the change
- Reinforcement — to make the change stick

FIGURE 5.2
The unfreeze/change/refreeze model can help you manage change.

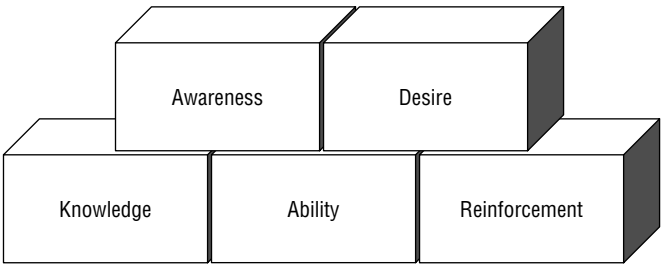


FIGURE 5.3
The ADKAR model is composed of five primary building blocks.

Putting Change Management into Practice

Now that you have a better understanding of the theories behind change management, we'll show you how to put those theories into practice. We'll focus on the role management plays in the process and the skills you need to manage change.

Understanding Management's Role

Large oceangoing ships have a rudder that allows the captain to steer the ship in the direction he or she wants to go. The rudder is not very big, but it can move a massive vessel in the desired direction.

Much like a rudder on a large ship, the management staff of an organization is comparatively small, but they can steer the entire organization in the direction they want it to go. This means upper management will shape the approach taken to change management in the organization. They do this by asking three types of questions about the proposed change:

Why *Why* questions are the foundation of change, and usually the first to be posed. Management will want to know things like:

- Why do we need a new reporting system?
- Why are we going with SQL Server Reporting Services?
- Why do we need it now?

What *What* questions focus on the output of the process. For example, management may want to know:

- What will change about the way we access and use reports?
- What milestones will signify success?
- What time, resources, and funds do we need to effect the change?

How *How* questions generally look at the required change from a mechanical perspective:

- How will we schedule the work?
- How will we roll out the new software?
- How will we train the users?

Different management teams will focus on different types of questions. Technical management usually focuses on the ''how'' questions. Management in business control departments, such as accounting and finance, usually focus on the ''what'' questions. Upper management focuses primarily on the ''why'' questions.

Once management understands the change fully, they can steer the company in the direction they want to go. To do that, management identifies trends in the organization as a whole and in each of the individual departments.

They may see, for instance, that the organization as a whole is adapting well to the new system, but the Maintenance department is complaining about the changes. Management will then find out why that department is having trouble and assign someone to work on the issue. It may be that the Maintenance department needs more training; they may want to be involved in some of the project status meetings; they may need someone to respond to some technical requests they've submitted. In any case, management should see to it that the department is taken care of.

If management sees that the entire organization is unhappy with the change, they may need to take more dramatic steps. This could translate into more training for the management staff in each department. They may decide to send a companywide e-mail on a regular basis to keep everyone updated on progress. They may decide to postpone the project until a better time. The important thing is to make sure that the majority of those in the organization agree with the needed change.

Necessary Skills for You

As with any task, you need some important skills to manage change. These are what we call ''soft skills'' because they primarily deal with working with people (as opposed to ''hard skills,'' which are the skills necessary to perform your core duties at work). You will need these skills to manage change effectively.

Political Skills

Corporate politics can be defined in a few different ways. First, it is the competition between groups or individuals for authority and leadership. It can also be defined as the total complex of relations between people in an organization. By these definitions corporate politics exist in almost every organization on earth, so knowing how to deal with politics is an essential skill.

To effectively deal with corporate politics during the life of your project, you need to understand the types of power that individuals wield, and hope to wield, in an organization. There are two main types: formal and informal power.

Formal power is conferred by the organization and usually comes with position. For example, the vice president of information technology has more authority over the IT department than a senior database administrator (DBA). When someone with formal power makes a request, it is prudent on your part to handle that request seriously.

Informal power is earned by those who wield it. A good example is a variant of informal power called expert power. When an individual in the organization is seen as an expert on a particular subject and people go to that person for advice, then that person has expert power. They can choose to give advice — or not. They can choose to give accurate advice — or not. They have the power to decide whom they want to help and whom they do not.

You will likely have some measure of formal power during the life of your project. You will have the authority to make some budget decisions, set deadlines, purchase hardware and software, and the like. You can then wield this formal power to help manage change.

You will also need to earn some degree of informal power to manage change during your project. You want people to see you as an expert in Reporting Services, you want them to come to you for guidance on the project, and you want them to follow that advice. There are a few simple tips to gain informal power:

- Cooperate with others; do favors for them whenever you can.
- Everyone wants some measure of authority, so delegate some of the more appealing assignments to others on the project.
- Compliment others on their accomplishments, and do not even mention your own accomplishments when doing so.
- Don't ask others to do what you are not willing to do yourself; get into the trenches and help with the dirty work when necessary.
- Be present at all of the meetings, no matter how boring they may seem.
- Be on time for meetings, especially if you called them.
- Be open, honest, and friendly with your coworkers.
- Be a part of the team, not just the leader.

Analytical Skills

Analysis is the action of taking something apart in order to study and understand it. Having analytical skills, then, means that you are able to break down a problem and come up with a plan of action to solve it. You are going to need analytical skills because a number of problems are going to arise during your project.

The challenge is that there is no definitive list of problems that could arise, so there are no predefined solutions to present to you here. You must be able to define the problems and develop solutions on your own. The most effective way to accomplish this goal is to ask questions about each problem. For example, suppose one of the department heads says, ''I'm not happy with the way this new system is shaping up.'' This is obviously a problem that you need to work on. You need to start by asking some questions to define the problem. It may flow like this:

You: Why are you unhappy with the system?

Department Head: It doesn't do everything we need it to do.

You: What seems to be missing?

Department Head: We need to open the reports in Excel so we can do some calculations on them.

You: I can show you how to save reports in Excel format and open them locally; will that help?

Department Head: That will be great! That's exactly what we need.

Defining the problem involves asking the right questions. Again, there is no comprehensive list of the ''right'' questions to ask for each problem, but you can come up with questions on your own by using the journalist's mantra:

- Who
- What
- When
- Where
- Why
- How

Asking these types of questions will get you to the root of any problem that may arise and then help you solve it.

Business Skills

You do not have to be a management guru to manage change in an organization, but you do need a basic understanding of how a business is run. This includes knowing how the business is managed, how it is organized, how it is financed, and so on.

We discussed these skills in detail in Chapter 2, so we won't rehash them here. We will offer a brief overview of what you should know to manage change from a business perspective.

First, identify the business reasons for implementing the system in the first place. The way to get that information is by sitting down with upper management (or as close as you can get) and

asking them why they have decided to move forward. Understanding these reasons will help you defend them later on.

Second, become familiar with the current business model, including the organizational structure, geographic scope, and business processes. This is necessary because you may well need to manage change at every level of the organization at every location.

People Skills

Without employees a business cannot function, which means that people skills are at the heart of effective change management. You must do your best to understand the people in your organization and see them as colleagues.

As we discussed in Chapter 4, this takes some effort on your part. Remember that people have different backgrounds, beliefs, native tongues, and likes and dislikes. You do not have to become a psychoanalyst, but try to take these human aspects into account in your interactions. There are some basic people skills you must develop to manage change.

First, you must learn to see the project through your users' eyes, because not everyone sees a project in the same way. You may view the project from a technical perspective, so you know that technical problems can be overcome fairly quickly. Marketing does not have that vantage point, so if you try to see the project through their eyes you will understand why they might be nervous when technical setbacks occur.

Also, it is important to remember that the users and coworkers in your company are people, not equipment. Remember to show courtesy and respect for your coworkers. Do not belittle them if they do not understand the new system. Thank them for any assistance they provide. Common courtesy can go a long way in getting people to accept change in an organization.

System Skills

As a technical professional, you may be thinking of computer system skills that need to be mastered during the life of the project. It is true that there are a number of such skills, but that is not what we are discussing here. Think of systems from a broader perspective.

According to General Systems Theory (GST), a system is a regularly interacting group of activities or parts that, when combined, form a new whole or system. For example, when a user works with the new reporting system, the two combine to form a new system. In other words, the whole is greater than the sum of the parts.

To simplify the theory a bit, you need to know how the users are going to work with the reporting system to form that new whole. That new whole is going to have properties that you would not expect by looking at the two individual parts separately. For example, you may have a report that shows sales data for the past month, and you may have a salesperson who uses that report regularly. When the salesperson uses the data from the report, they may change their selling tactics and start pitching a new product. This can affect other areas of the company, such as accounting and production.

This is an especially important skill to acquire if you are introducing new reports into the system as part of the project. Consider how your users will use the reports and how that might introduce change into the organization.

General Systems Theory was proposed and solidified in the late 1940s and early 1950s by Ludwig von Bertalanffy and others (`http://en.wikipedia.org/wiki/Ludwig_von_Bertalanffy`).

Change-Management Strategies

In 1984, Kenneth Benne and Robert Chin authored an article on change management strategy (Chin, Robert and Benne, Kenneth D. ''General Strategies for Effecting Changes in Human Systems.'' *The Planning of Change 4th Edition*. 1984). In their article, the authors summarized various approaches to change management in a framework of three general strategies. This framework still serves as a guide today and will help you choose a viable strategy for change. These are the three basic strategies for change proposed by Benne and Chin that we will discuss.

Empirical-Rational

The base assumption of this strategy is that people are rational. This means that, when someone is presented with a change that is in their self-interest, they will accept that change as a means of attaining that interest.

In practice, the strategy works this way: you (the change agent) put forward a change that you believe is in your users' best interest — in this case, a new reporting system. As you propose the change, you present rational arguments in favor of the change, pointing out the links between the change and the users' best interests. The logical outcome is that the users should agree with the rational arguments and see that the change is going to be in their best interest.

The key to this strategy is information — and lots of it. Users should have all the information they need to decide that the change you are asking them to make is in their best interest. You should also present this information in a way that makes sense. If the users do not understand the information you are sharing, they can hardly use it to make a decision.

Normative-Reeducative

While the empirical-rational strategy is based on rational thought as the key to effecting change, the normative-reeducative strategy is based primarily on feelings and values. The users must feel that the current system is not satisfactory and the change that you are proposing must feel like the right method for rectifying those inadequacies.

The basic assumption in this strategy is that intelligence is social rather than rational. In other words, change can only be effected when the majority of systems interact (systems as described earlier in this chapter, not as machines).

This strategy goes beyond personal feelings of what is right; it extends to include organizational values. Empirical-rational strategy assumes that people will embrace change whether or not it is aligned with organizational values because it is rational to do so. Normative-reeducative strategy states that users will not embrace change if it does not align with organizational values, even if it is rational.

Power-Coercive

As the name implies, this strategy focuses on legitimate power as a means to effect change. This approach has similarities to the other two approaches. For instance, the empirical-rational approach states that users will accept change because they have the knowledge to make the decision to do so. The power-coercive approach states that the people who have the knowledge also have the power to make changes in the organization. The normative-reeductaive approach to change involves giving people a sense of limited personal power (feeling that they are doing the right thing). In the power-coercive approach, users are told that they are doing the right thing by changing.

In essence, this approach states that users will accept change because management said that it is best for the organization. Interestingly, there is a reverse scenario, too. Users can exercise power to

make management change in some way. Unions use this power regularly to convince employers to change the way they function.

Choosing a Strategy

So, how do you choose the strategy that will work best for you? There is no right or wrong strategy; you can choose any of them. In fact, you will probably end up combining them into a larger master strategy that works across the entire organization; some departments may require a power-coercive strategy while others work best with an empirical-rational approach. You will have to get to know your user base to understand which approach to use. We discuss a few guidelines that can help you choose.

First, think about how much resistance there is to change in your organization. If a high degree of resistance exists, you may need to use the power-coercive method to bring everyone in line with the company's objective. Also, if you do not have a lot of time to complete your project, you may decide to use the power-coercive method.

Most organizations require a different strategy for different user groups. If you are dealing with a group of users who are technically savvy, such as IT, then you should use the empirical-rational approach. If you are dealing with a management group, the normative-reeducative approach may be best.

Managing Ongoing Change

After you have convinced everyone that Reporting Services is a good move for the company, they have accepted the change, and the move has been made, your work is still not done. You have to manage ongoing change in your reporting system.

This is important because if you do not manage ongoing change, your system could be brought to a screeching halt because of a flawed update. Suppose, for example, that you need to change the schema for a database that is being used as a data source for one of the reports that accounting uses to perform payroll every week. If you do not have a change-management system in place and the schema change breaks the report, you will not know how to roll back to the previous version of the database and payroll may be delayed.

To manage ongoing change, you need to have a change-management system in place. This may be something as simple as a form that must be filled out and approved by a manager, or something as complex as a corporate change management board. Either way, you must have this scheme in place by the time your new reporting system goes into production.

If you do not have a change-management request form, you can design one of your own. It should contain information such as:

- The date of the proposed change
- The system(s) affected by the change
- A detailed description of the proposed change
- A detailed rollout plan
- A detailed rollback plan in case the change causes problems
- A list of users to notify about the change

Make sure that your system administrators and developers use this change request form every time they make a change to the system. You may decide that they do not need to use this when they

are making a minor change, but they should certainly use it for major changes to the system. The rule of thumb is that if the change can possibly cause problems in the system, create a change request.

Real World Scenario

Case Study: Chocolaty Goodness Candies

Chocolaty Goodness Candies had a change-management system in place. They would fill out a paper form with pertinent information about the change and submit that form to the IT manager. If the manager approved of the change, he would sign off on the form and file it in a cabinet.

After filling out a few of these forms, the consultant i realized that i didn't like filling out forms on paper when i could fill out electronic forms. So i duplicated the paper form in InfoPath and stored the form in a SharePoint document library. I made a few programmatic changes to the InfoPath form so that the manager could approve or reject the change using a button on the form. When the form was processed at any stage, a Windows Service sent an e-mail to the appropriate user to let them know about the progress. For example, the manager would get an e-mail when a new change request had been submitted, users would get e-mails about acceptance status, and end users would be notified of pending changes.

This strategy saved the company from unmitigated disaster on a few occasions — for example, when they needed to apply a service pack and subsequent updates to the load-balanced web servers. All of the updates installed without a hitch, but when they were done, the web servers would not communicate with the database servers. Thanks to a detailed rollback plan, they were able to bring the web servers back on line fairly quickly. If they had not had a change request with a detailed rollback plan, they would have had a much harder time recovering the servers.

The Bottom Line

Understand the theory of change. To effectively manage change for your organization, you must understand some of the theories behind change.

Master It What are some theories of change and what do they consist of?

Understand the skills needed to effect change. Knowing the theories of change is not enough to make change happen. You need to master some skills in order to effect change.

Master It What are some of the skills that you need to acquire to manage change in your organization?

Choose a change-management strategy. Once you know what change is all about and you have the skills to manage it, you need to choose a strategy for managing change throughout your project.

Master It What are three main strategies for managing change in an organization?

Manage ongoing change. After the project is complete and the changes have all been made, you must continue to manage ongoing change. If you do not manage changes to the system, your entire reporting system could crash — which would cause you to lose user confidence.

Master It What should you include in a change request form?

Chapter 6

Business Intelligence and Reporting

Organizations collect vast amounts of information in the course of doing business. They collect information about their customers, sales, products, promotions — the list goes on. All of that information is stored somewhere; be it Access, Excel, SQL Server, Oracle, or what have you, the data is saved in an electronic format. Management wants, and needs, to study that data to make decisions about the business.

For years, people were looking for a way to define the processes involved in saving and studying the copious amounts of data stored during business transactions. In 1989, Howard Dresner coined the term ''business intelligence'' (BI). This term refers to the applications and technologies that are used to gather, store, retrieve, and analyze all of the data used in making business decisions.

BI systems give managers a detailed view of their business. By tracking various metrics about sales, production quotas, internal operations, and the like, managers can make more informed decisions about what to do in their business.

SQL Server has four main components that fit into the BI framework:

- The database engine
- Analysis Services
- Reporting Services
- Business Intelligence Development Studio

The database engine allows you to store data about anything you need and then retrieve that data later for study. Analysis Services provides data-mining services, which help you get a more comprehensive picture of your data than standard SQL queries provide. Reporting Services is the tool that allows managers to see all this information so that they can make decisions. The Business Intelligence Development Studio is the tool that your developers will use to create new reports and report models.

In this chapter you will:

- Gain a better understanding of OLAP
- Gain a better understanding of data mining
- Learn how to base a report on a cube or a data-mining model

Business Intelligence in SQL Server 2005

Why do you need to use the business intelligence tools that come with SQL Server 2005? Let's consider an example: suppose that you work for a company that manufactures and sells clothing. You probably have a number of databases that store information on sales for each of your stores,

production quotas, raw materials, and the like. You could use standard SELECT statements to query the database for sales figures for a given month, whether production quotas have been met, what raw materials are being used, and so on. These smaller queries are called online transaction processing (OLTP) queries, and they are fine if you need to see only a small amount of data.

The catch is that management usually looks for more data than OLTP queries can provide in a timely manner. Suppose that management wants to know the sales figures for women's hats in Michigan for the previous March and they want to correlate that information with the raw materials that were received so that they can order enough materials for next March. At the same time, they want to know whether the production quotas were met for that timeframe so they know whether they have to hire additional workers to meet the quotas next March. You could write an OLTP transaction to return this data, but that is not usually feasible. The database involved in such queries may contain millions of records and they may be dispersed on several different servers throughout the organization. An OLTP query could take a long time to return a result.

To speed things up, SQL Server 2005 comes with Analysis Services, which has two important functions: OLAP and data mining.

NOTE

This chapter presents a brief overview of Analysis Services and business intelligence in general. For a more detailed explanation, see *Mastering SQL Server 2005* (Sybex, 2006).

Online Analytical Processing

The principle behind online analytical processing (OLAP) is simple: trade extra storage space for accelerated processing. To accomplish this, SQL Server 2005 provides Analysis Services, which allows you to create and manage multidimensional structures. These structures contain data collected and aggregated from other sources. So, instead of accessing raw data from a database and performing complex calculations on the data to get the results you need, you can access pre-aggregated data from an OLAP cube, which provides results much faster. It also features data-mining capabilities, which allow you to create data models used for finding patterns in your data (more on that later). To understand how OLAP works, you need to know some basic terminology.

CUBES

The basic unit of storage in Analysis Services, *cubes* are collections of data that are based on tables and views stored in a database. Cubes are organized into dimensions, which are stored in dimension tables, and measures, which are stored in fact tables.

Dimensions are used to organize data by topics of interest. For example, you may have a sales table that you would like to organize by date of sale, or a customer table that you would like to summarize by city. Dimensions should be hierarchically organized — for instance, cities rolling up into states and states rolling up into countries. These dimensions are stored in dimension tables.

Measures are pieces of data that you want summarized by dimension, such as sales totals or annual salaries. Anything that can be aggregated (usually numeric) can be used as a measure. Measures are stored in fact tables.

Fact tables and dimension tables are closely related, which makes sense considering that measures are aggregated by dimensions. These tables are related in a cube using a schema. Among the many schemas available, the star and snowflake schemas are the most common.

In a star schema, as shown in Figure 6.1, every dimension table is related directly to the fact table. In a snowflake schema, as Figure 6.2 shows, some dimension tables are related to other dimension tables, which are related to the fact table.

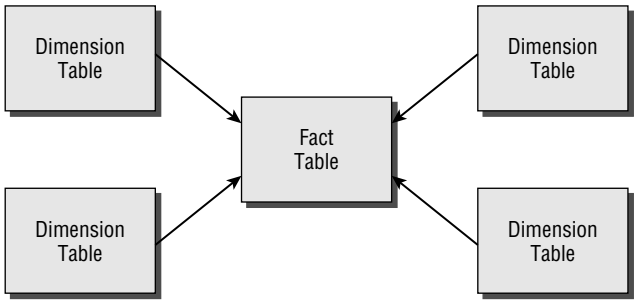


FIGURE 6.1
Every dimension table is related directly to the fact table in a star schema.

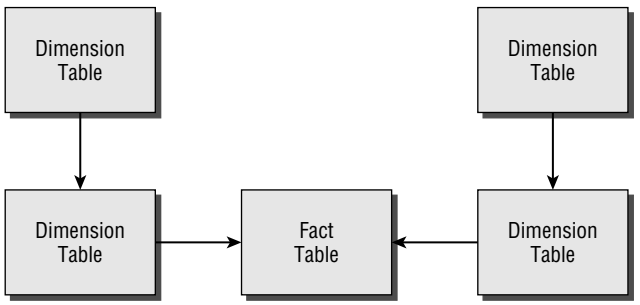


FIGURE 6.2
Dimension tables may be related to the fact table through other dimension tables in a snowflake schema.

It is easier and faster to query all of this data when it is aggregated and stored in one place that it would be if the data were scattered about the enterprise in its pre-aggregated state.

Storage Settings

The detail and aggregated data that makes up a cube can be stored in the cube itself or in the relational database that houses the dimension and fact tables. Where these pieces of data are stored depends on the storage setting you choose when creating your cube.

ROLAP With relational OLAP (ROLAP), the only object residing in Analysis Services storage is the cube definition. The detail data is stored in its original location on the database server, and the aggregate data is stored in a separate set of tables. This setting works well for large data sets that are not queried frequently.

MOLAP Multidimensional OLAP (MOLAP) copies all of the detail and aggregate data to the analysis server for storage. This takes up more storage space than ROLAP, and it is also faster because the data is more readily available and pre-aggregated. You can choose among five subtypes of MOLAP: standard, low latency, medium latency, high latency, automatic, and scheduled.

HOLAP Hybrid OLAP (HOLAP), as the name implies, is a mix of MOLAP and ROLAP. With this setting, the detail data remains on the database server and the aggregated data is copied to the Analysis Services server for storage.

NOTE

For more on OLAP storage settings, refer to the topic "Choosing a Standard Storage Setting" in SQL Server Books Online.

Multidimensional Expressions

The language that you use to access data in a SQL Server database is Transact-SQL (TSQL). To access data through Analysis Services, you need to use Multidimensional Expressions (MDX). You can also use DMX, which we will discuss later in this chapter.

MDX is a statement-based scripting language that looks and feels a lot like TSQL, but it is a completely separate language that is used solely for working with cubes in Analysis Services. MDX provides these language features:

- Data definition language (DDL) statements used to create, drop, and modify multidimensional objects
- Data manipulation language (DML) statements for retrieving and manipulating data from multidimensional objects
- Scripting language statements for managing scope, context, and control of flow within MDX scripts
- User-defined functions for extending MDX

The MDX language has a number of components that you can use, so the queries can become quite complex. These queries are useful to you because you can use MDX with Reporting Services to create multidimensional reports.

Data Mining

Analysis Services cubes are designed to give you quick access to aggregated data. The question now is, what will you do with that data?

Management needs to make business decisions based on the data in those cubes. To make business decisions, management looks for patterns in the data. For example, management may need to know how many men's leather coats were sold in the northeastern United States last November so they can make preparations for the coming November. They may also look at production quotas, sales by store, materials usage in manufacturing, and so forth. All of this information forms a pattern, or series of patterns, that they can use in making decisions. The process of looking for such patterns in your data is called *data mining*.

By themselves, cubes do not show patterns; they only store data that managers can use to look for patterns, or mine the data, manually. You can automate this process by using the built-in data mining functionality that Analysis Services provides.

When you create a data-mining solution in Analysis Services, you first need to create a data-mining model that defines the business problem you are trying to solve. Once you have created the model, you can train the model by running your data through a mathematical algorithm that generates a mathematical model of the data. At this point the model is ready for use.

You can query the data-mining model using the Data Mining Extensions (DMX) query language. Like MDX, DMX is a full-featured query language that includes DML and DDL statements, which you can use to work with data-mining models.

Once you have created and trained a model, you can use the result of a DMX query as the basis for a Reporting Services report. Before you can create a report based on MDX or DMX, you must install Analysis Services and configure a cube.

Installing and Configuring Analysis Services

Now that you have a high-level overview of what Analysis Services is used for, you probably want to start using it with Reporting Services to see what they can do together. First, you must install Analysis Services and configure the sample data. Then you have to create an Analysis Services project and create a cube in that project.

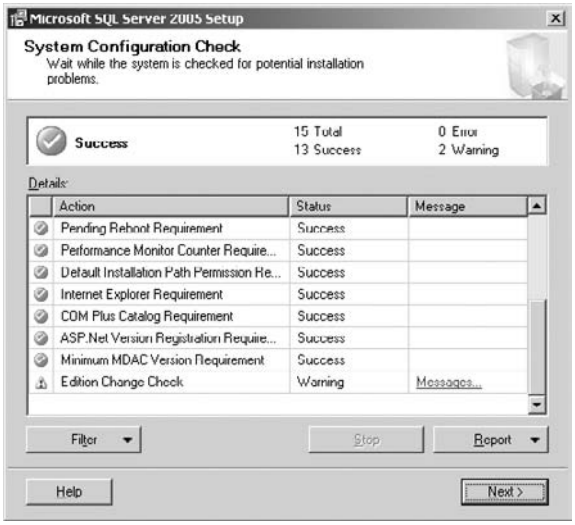
Installing Analysis Services

Follow these steps to install Analysis Services with some sample data:

NOTE

In the following steps, we assume that you do not have Analysis Services installed. If you do have Analysis Services installed, you may still need to install the sample data. To do so, follow these steps with the exception of step 12.

1. Log on to your server as an administrator.
2. Open Add/Remove Programs from Control Panel.
3. Look for SQL Server 2005 in the list of installed programs and click Change.
4. At the top of the Component Selection dialog box, click the link labeled ''To install a new component, click here.''
5. Locate the `Setup.exe` file on your installation media and click OK.
6. In the EULA dialog box, check the acceptance box and click Next.
7. After the prerequisites have been installed, click Next.
8. On the Welcome screen, click Next.
9. Click Next on the Configuration Check screen. You may see a warning message regarding the Edition Change Check. This is expected if you are running SQL Server Service Pack 1; the warning can be safely ignored.

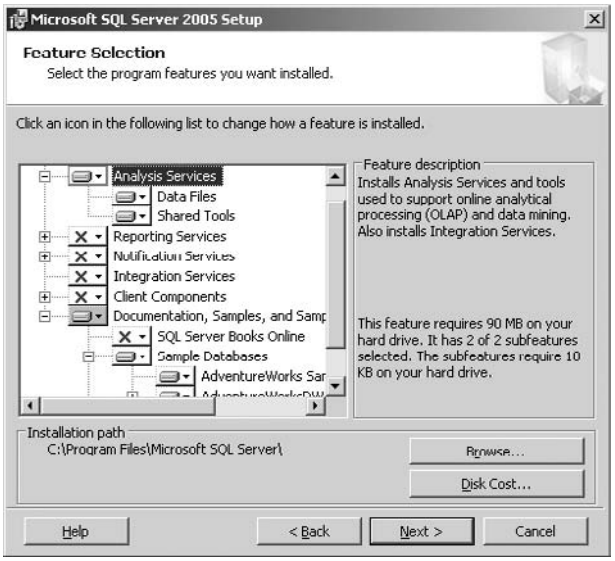


10. Fill in the information on the Registration Information screen and click Next.
11. On the Components To Install screen, click the Advanced button at the bottom right of the screen.

12. Click the button next to Analysis Services and select ''Entire feature will be installed on local hard drive.''

13. Expand the Documentation, Samples, and Sample databases.

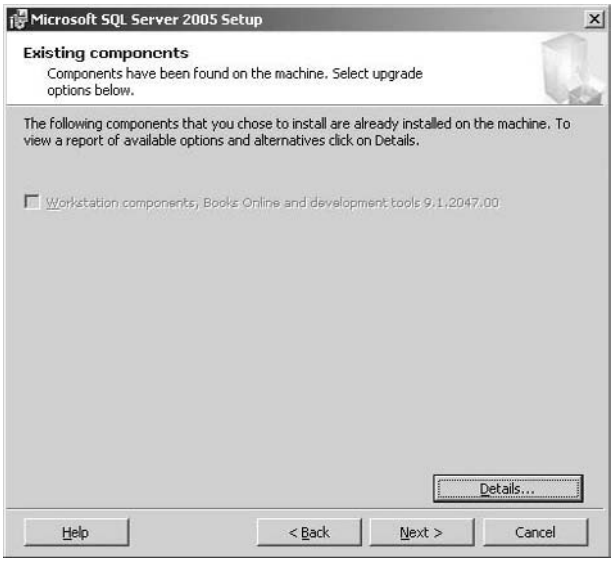
14. Click the button next to Sample Databases and select ''Entire feature will be installed on local hard drive.''



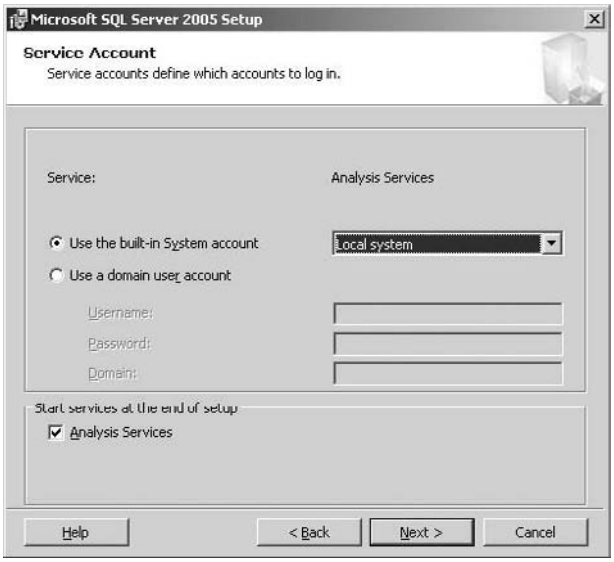
15. Click Next to continue.

16. Make sure the Default instance is selected on the Instance Name screen and click Next.

17. If you are running SQL Server 2005 Service Pack 1, you may be presented with the Existing Components screen informing you that you cannot upgrade some components. This is expected; click Next.



18. On the Service account screen, select the ''Use the Built-in System Account option'' and select Local System from the drop-down list of accounts. Then click Next.



> **NOTE**
>
> The Local System account is not recommended for production systems; it is usually best to use a domain account. We are only using it for the purposes of this exercise.

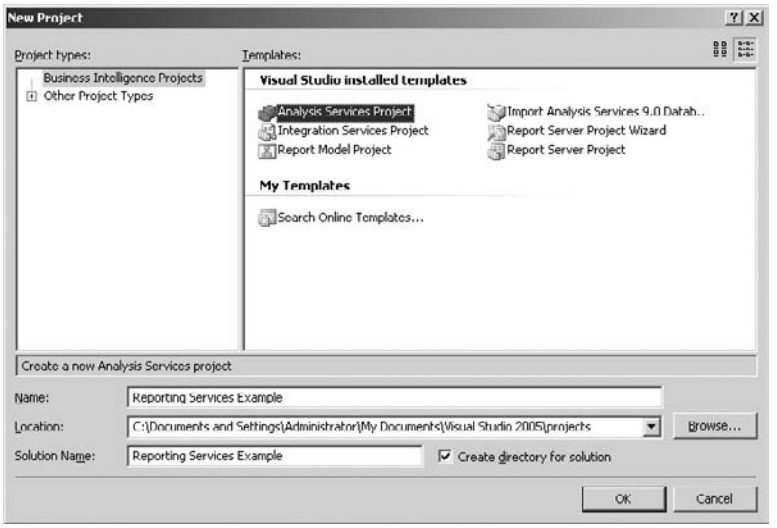
19. Accept the default options on the Collation Settings screen and click Next.
20. Click Install on the Ready To Install screen.
21. When setup is complete, click Finish.
22. After the installation is complete, be sure to reinstall SQL Server 2005 Service Pack 1 (or install it for the first time if you have not done so).

Creating and Configuring a Project

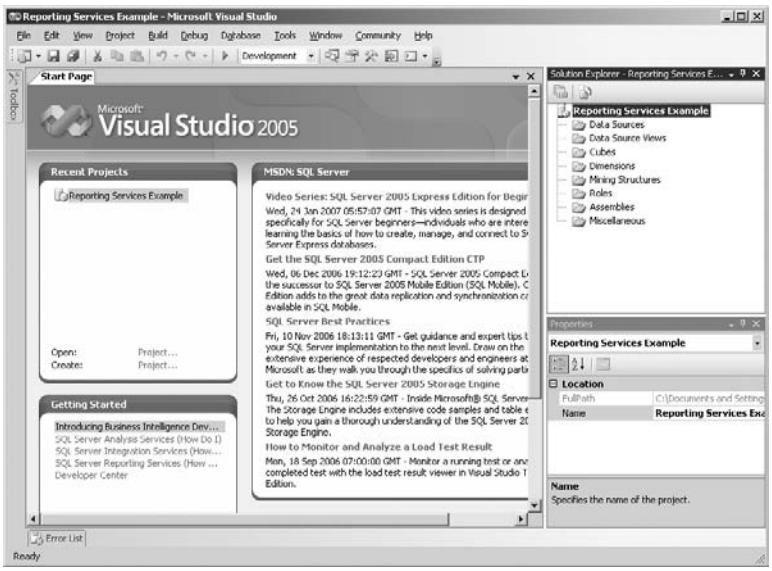
Now that you have Analysis Services installed and running, you need to create a project to work with. You create and manipulate Analysis Services projects in Business Intelligence Development Studio (BIDS).

In the following steps, you will create an empty Analysis Services project:

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio to open BIDS.
2. Select File ➢ New ➢ Project.
3. Select Business Intelligence Projects from the Project Types list.
4. Select Analysis Services Project from the Templates list.
5. Type **Reporting Services Example** in the Name text box (note that the Solution Name box is filled in for you).

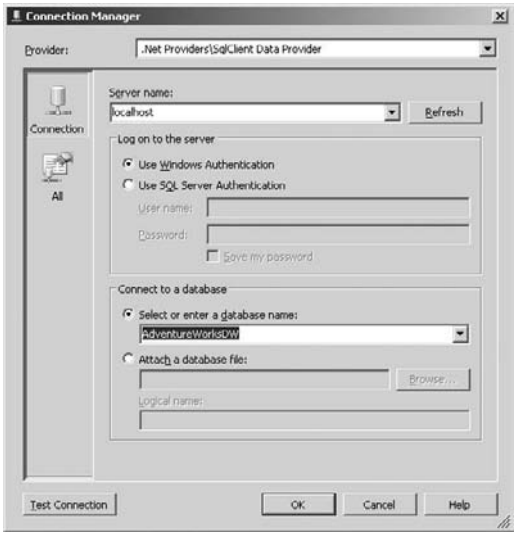


6. Click OK to create the project.

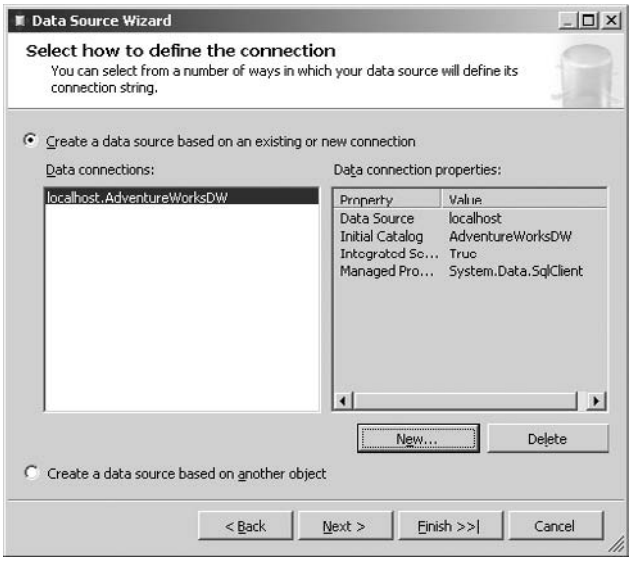


As you can see, the project has no data when it is first created, so you need to populate it. You need to connect the project to a database to populate it, so next you will create a data source for the project that connects to the AdventureWorks Data Warehouse (DW) sample database:

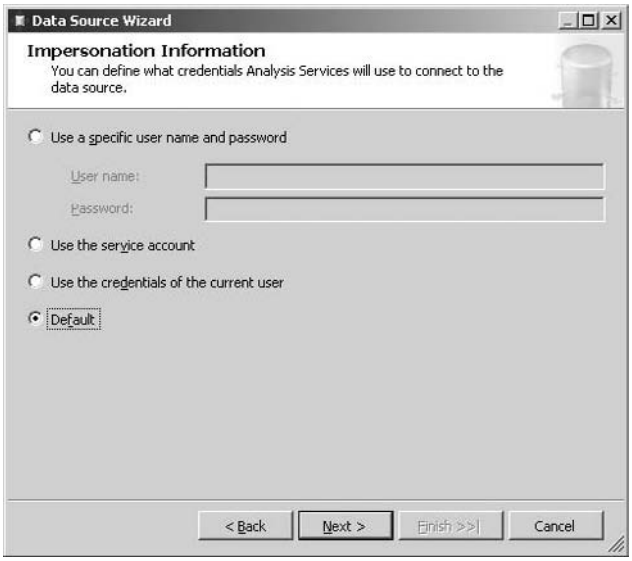
1. In the Solution Explorer, right-click the Data Sources folder and click New Data Source to start the New Data Source Wizard.
2. On the Welcome screen, click Next.
3. On the Select How To Define The Connection screen, click New.
4. In the Provider list box, select SqlClient Data Provider under `.NET` Providers and click OK.
5. Type **localhost** in the Server Name box.



6. In the ''Connect to a database'' section, make sure the ''Select or enter a new database name'' option is selected and choose AdventureWorksDW from the drop-down list.
7. Click OK to create the connection.



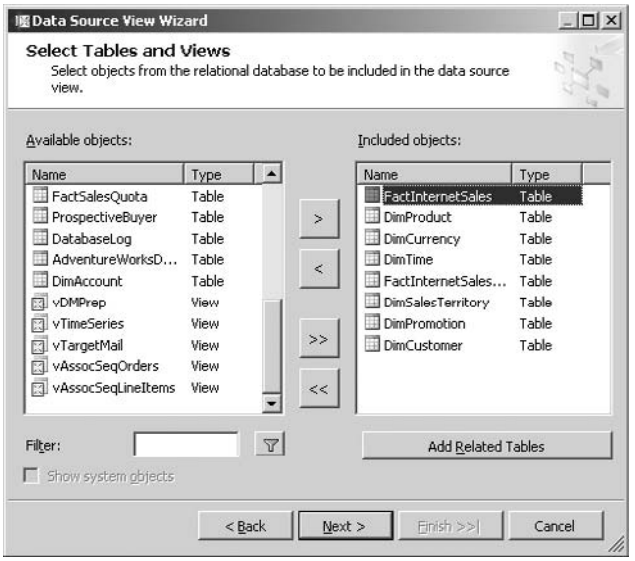
8. Click Next to continue.
9. On the Impersonation Information page, select the Default option and click Next.



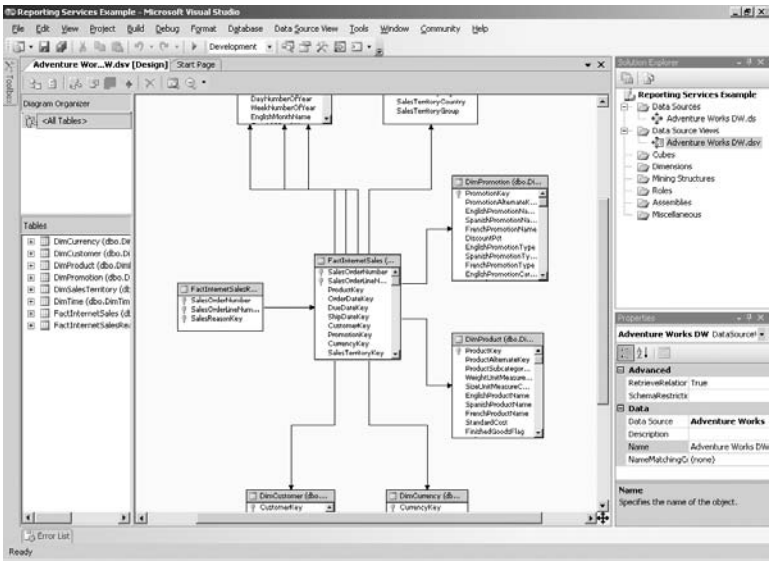
10. On the Completing The Wizard screen, click Finish.

At this point, you need a data source view to work with the data source. A data source view is the logical representation of the underlying data source. It contains the metadata and schema definitions used by cubes and mining structures. Let's create a data source view now:

1. In the Solution Explorer, right-click the Data Source Views folder and click New Data Source to start the New Data Source View Wizard.
2. On the Welcome screen, click Next.
3. On the Select A Data Source screen, make sure the Adventure Works DW data source is selected and click Next.
4. Select the FactInternetSales table in the Available Objects list and click the > button to add it to the Included Objects list.
5. Click the Add Related Tables button to add all of the tables related to FactInternetSales.



6. Click Next to continue.
7. On the Completing the Wizard screen, click Finish.
8. Your new data source view should now be open in the designer.



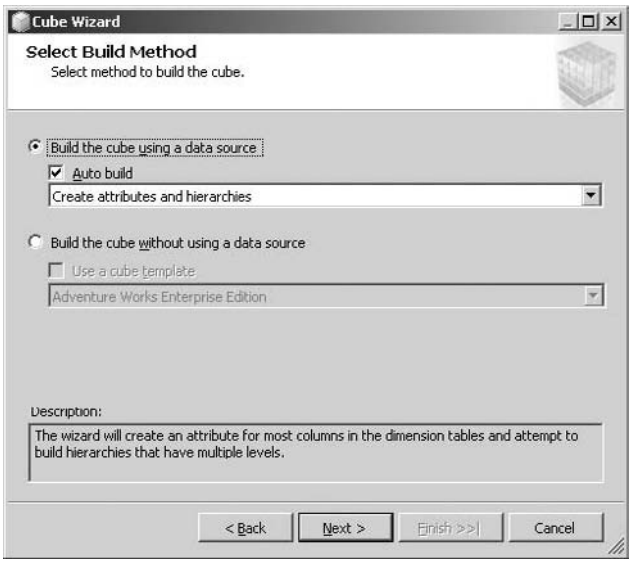
Now that you have a project with a data source and a data source view, you are ready to create a cube.

Creating a Cube

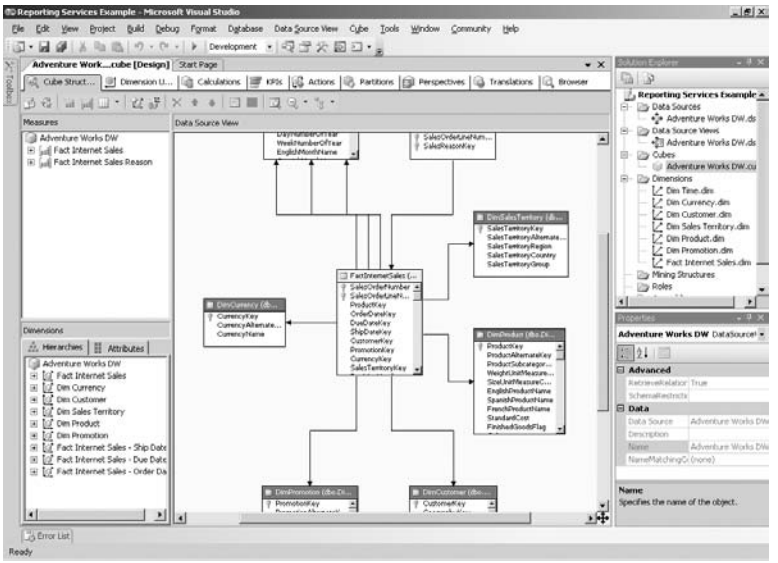
At this point, you are ready to create a cube based on the data source view. There are two ways to define a cube. The first method is to define several dimensions, independent of a cube, and then create a cube based on those dimensions.

The second method is to create a cube and all of its corresponding dimensions at the same time using the Cube Wizard. After you create the cube, you have to deploy it to the server so that you can access it through Reporting Services. Let's create a cube using the Cube Wizard and deploy it now:

1. In the Solution Explorer, right-click the Cubes folder and click New Cube to start the New Cube Wizard.
2. On the Welcome screen, click Next.
3. On the Select Build Method screen, make sure the Build The Cube Using A Data Source option is selected, the Auto Build option is checked, and the Create Attributes And Hierarchies option is selected from the drop-down list. Then click Next.



4. On the Select Data Source View screen, select the AdventureWorks DW data source view and click Next.
5. On the Detecting Fact and Dimensions screen, click Next after the detection process is complete.
6. On the Identify Fact And Dimension Tables screen, notice that two of the tables have been detected as fact tables; the rest are dimension tables. Accept these defaults and click Next.
7. Accept all of the defaults on the Select Measures screen and click Next.
8. On the Detecting Hierarchies screen, click Next after the detection process is complete.
9. Accept all of the defaults on the Review New Dimensions screen and click Next.
10. Click Finish on the Completing The Wizard screen.
11. When the wizard is complete, you will see a cube and several dimensions in the Solution Explorer.



12. Right-click the Reporting Services Example in the Solution Explorer and click Build to build the project.

13. Right-click the Reporting Services Example in the Solution Explorer and click Deploy to deploy the project.

You now have a cube in place and thus have the foundation you need to create a new report in Reporting Services based on an Analysis Services cube.

Creating a Report Based on a Cube

Creating a report based on an Analysis Services cube is not much different from creating a report based on any other data source. In the following steps, we'll use the Report Server Project Wizard in Business Intelligence Development Studio (BIDS) to create a report that shows you the total quantity ordered, sales amount, and freight amount for each product in the Reporting Services Example cube grouped by product name:

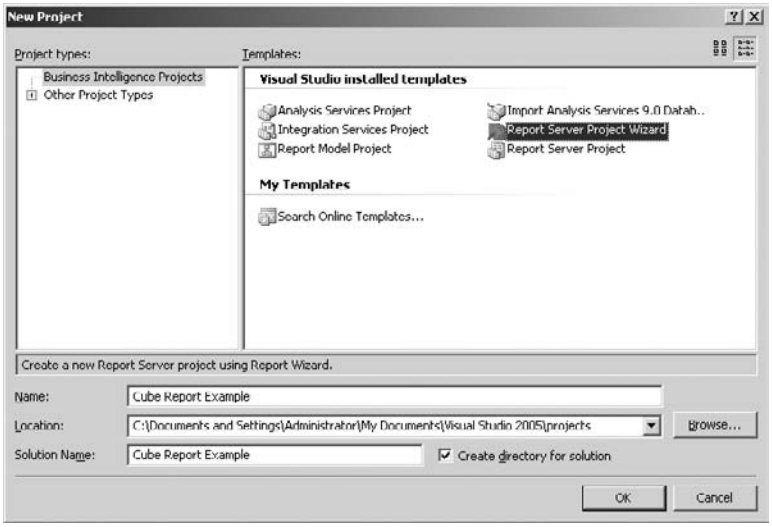
1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio to open BIDS.

2. Select File menu ➢ New ➢ Project.

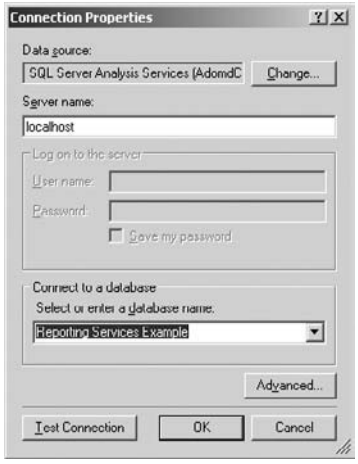
3. Select Business Intelligence Projects from the Project types list.

4. Select Report Server Project Wizard from the Templates list.

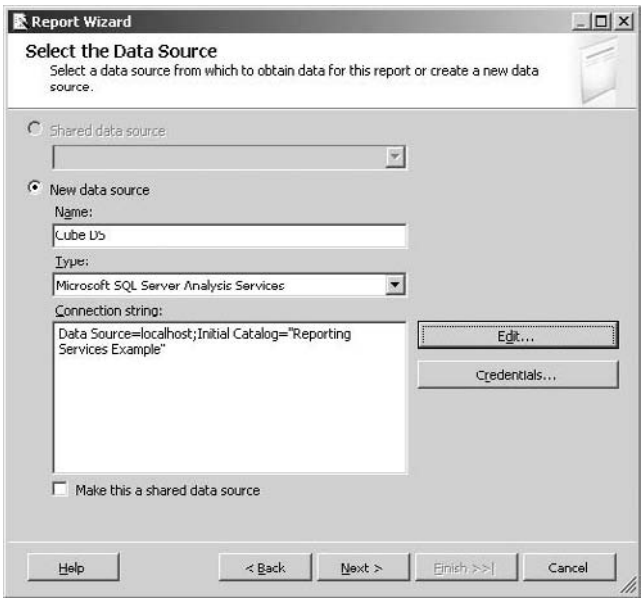
5. Type **Cube Report Example** in the Name text box (note that the Solution Name box is filled in for you).



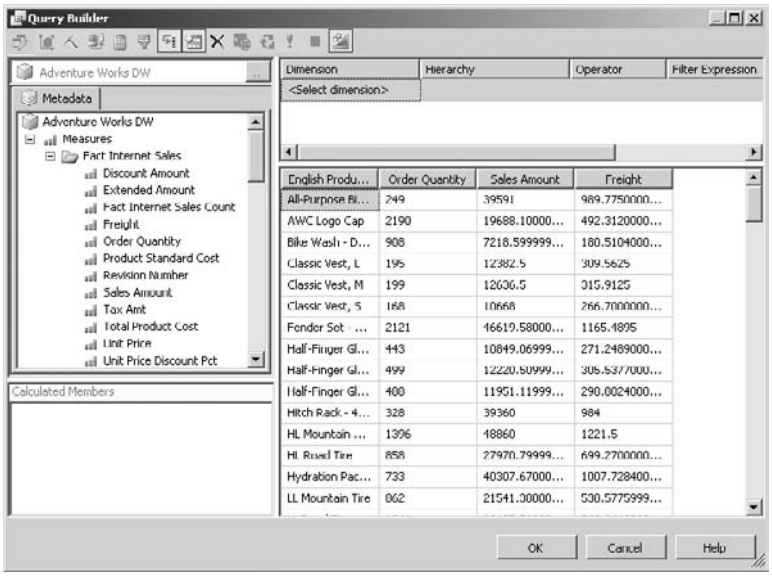
6. Click OK to start the wizard.
7. On the Welcome screen, click Next.
8. On the Select The Data Source screen, type **Cube DS** in the Name box.
9. Select Microsoft SQL Server Analysis Services from the Type drop-down list box.
10. Click the Edit button to create a connection string.
11. Type **localhost** in the Name box (or the name of the server Analysis Services is installed on if it is not your local server).
12. Select Reporting Services Example from the drop-down list box under ''Select or enter a database name.''



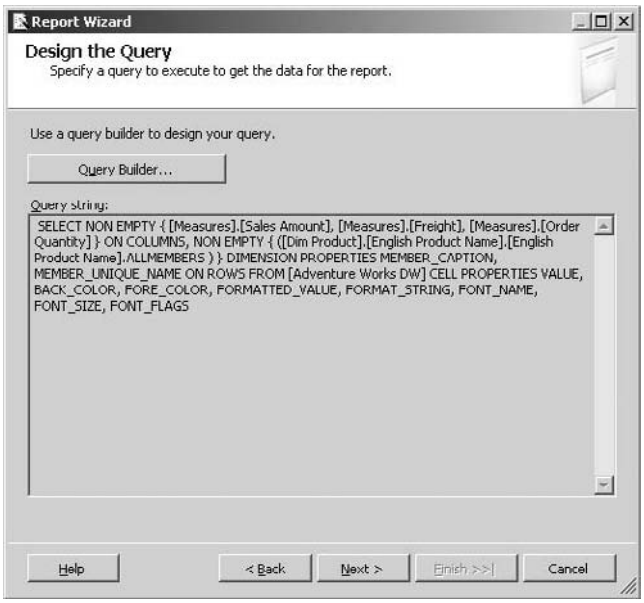
13. Click OK to return to the wizard.



14. Click Next to continue.
15. Click the Query Builder button at the top of the Design The Query screen.
16. In the Metadata pane, expand DimProduct.
17. Drag the English Product Name Dimension to the area labeled ''Drag levels or measures here'' to add to the query.
18. In the Metadata pane, expand Measures ➢ Fact Internet Sales.
19. Drag the Order Quantity measure to the query area.
20. Drag the Sales Amount and Freight measures to the query area.



21. Click OK to return to the wizard. Notice that the Query Builder has constructed an MDX language query for you.

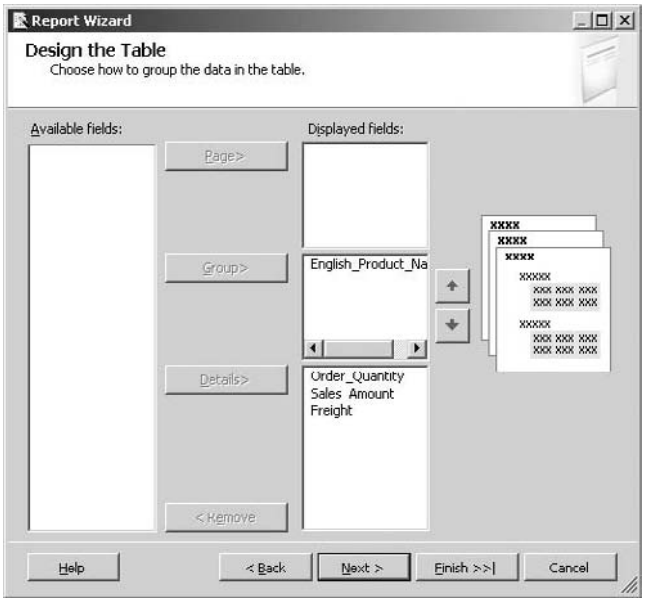


22. Click Next to continue.

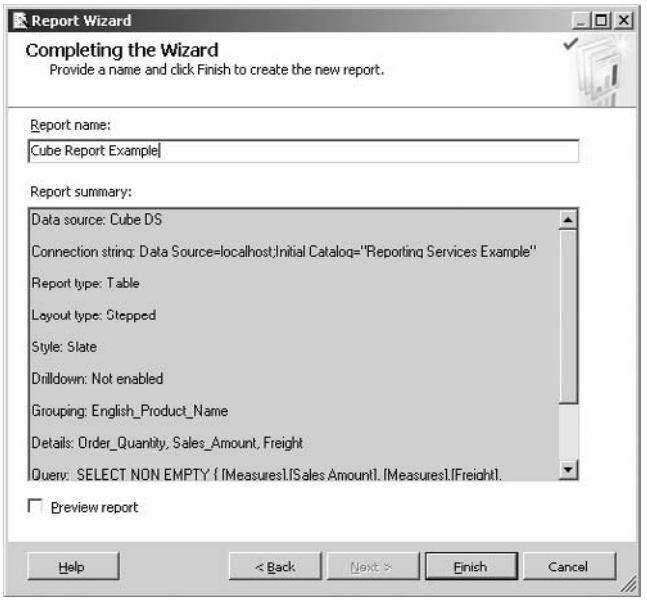
23. Select the Tabular option on the Select Report Type screen and click Next.

24. On the Design The Table screen, select the English Product Name field from the Available Fields list and click the Group button.

25. Add the remaining fields to the Details area by selecting them one at a time and clicking the Details button.



26. Click Next to continue.
27. Accept the defaults on the Choose The Table Layout screen and click Next.
28. Select a style on the Choose The Table Style screen and click Next.
29. Accept the defaults on the Choose The Deployment Location screen and click Next.
30. Type **Cube Report Example** in the Report Name box and click Finish to create the report.



That's all there is to it; you now have a report based on an Analysis Services cube. This particular report shows you the total quantity ordered, sales amount, and freight amount for each product in the cube grouped by product name.

The biggest difference between a report based on Analysis Services and one based on TSQL is the data pane. The data pane in a TSQL-based report looks like the query window used in Management Studio. The data pane in an Analysis Services–based report looks almost like the cube explorer in BIDS. Aside from that difference, the report works just like a TSQL-based report.

Real World Scenario

Case Study: Chocolaty Goodness Candies

CGC sells many different types of candies all over the world, which is a complicated process.

First, they need to purchase raw materials in the right quantities to make the different types of candy. This is a challenge because some candies require the same ingredients but in different amounts. For

example, there are about 15 different types of candy that all require the same red dye, but they all require a different amount.

CGC also needs to make sure that they have the right number of people to make the candy at the right time. For instance, CGC does not need as many people to make candy for the Valentine's Day season as they do for the Christmas season.

The executive staff needs accurate data to make decisions regarding labor and materials, and that data was scattered throughout the company in various databases. I could have created some complex OLTP queries to retrieve that data, but the queries would have been unacceptably slow. So instead, our team created an OLAP cube to aggregate all of the production data, which we used to create several reports to display production information. The executive staff was quite pleased with the result.

Creating a Report Based on a Data-Mining Model

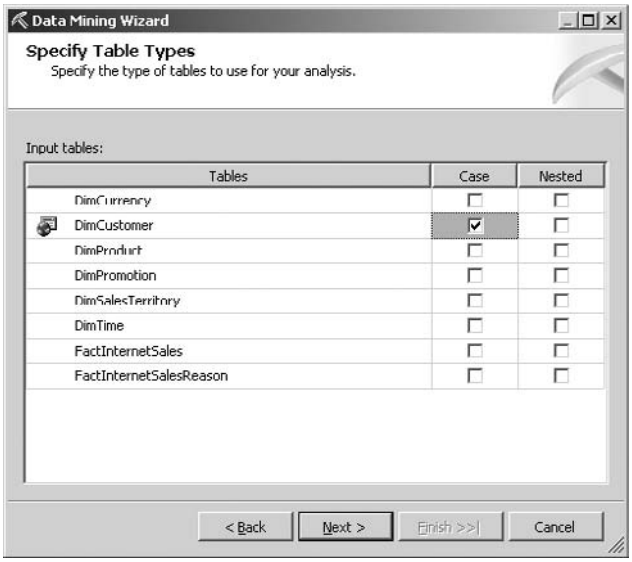
As previously stated, there is more to data analysis than just cubes. Managers look through the data to find patterns that will help them make accurate predictions about future business trends, which, in turn, help them make better business decisions. The process of scouring data to find patterns is called *data mining*.

SQL Server Analysis Services provides tools to automate the process of data mining to make the process easy for everyone involved. The topic of data mining is an involved one that could take up an entire book by itself, so I will not go into the details here. In this section, you will see how to create a basic data-mining model and then create a report based on the model.

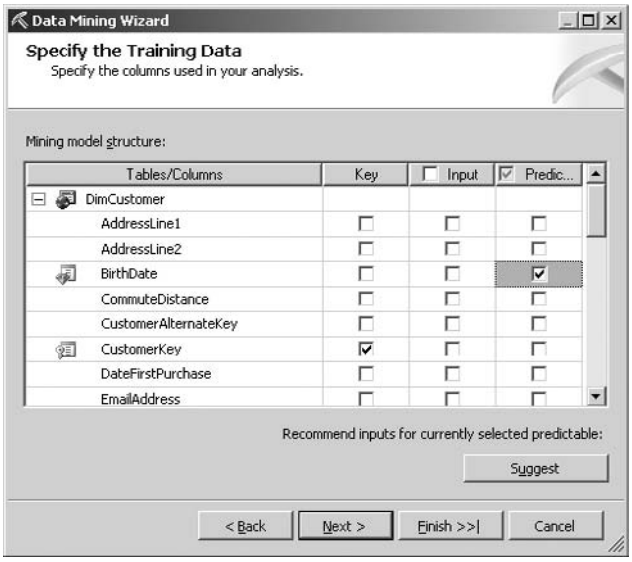
Creating a Data-Mining Model

Before you can create the report, you need to create a data-mining model using the Data Mining Wizard. Let's do that now:

1. If BIDS is not already open, click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio to open it.
2. Select File ➢ Open ➢ Project/Solution and open the Reporting Services Example project you created earlier in this chapter.
3. In the Solution Explorer, right-click the Mining Structures folder and click New Mining Structure to start the Data Mining Wizard.
4. On the Welcome screen, click Next.
5. On the Select The Definition Method screen, select the ''From existing relational database or data warehouse'' screen option and click Next.
6. Accept the default option of Microsoft Decision Trees on the Select The Data Mining Technique option and click Next.
7. On the Select Data Source View screen, click Next.
8. On the Specify Table Types screen, check the Case box for the DimCustomer table and click Next.



9. On the Specify The Training Data screen, you need to define what data will be used to train the model. To start, check the Predictable box next to the BirthDate column.

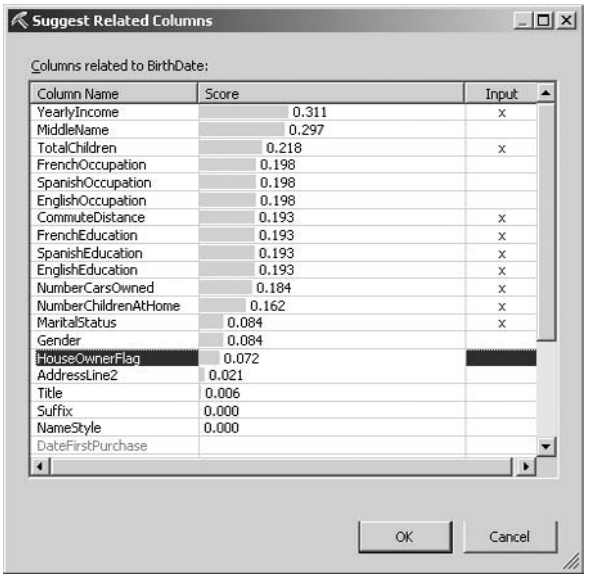


10. Click the Suggest button at the bottom of the screen to see what columns are related to the BirthDate column and what their correlation score is.

11. Uncheck the following columns by clicking the x in the Input column:

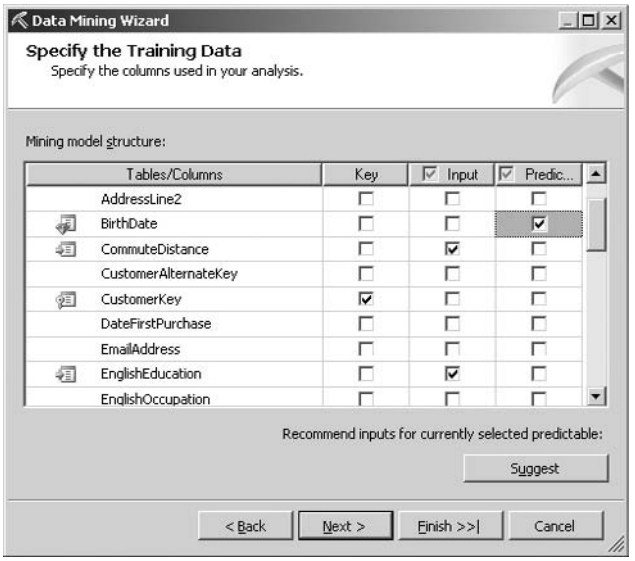
- MiddleName
- FrenchOccupation

- SpanishOccupation
- EnglishOccupation
- Gender
- HouseOwnerFlag

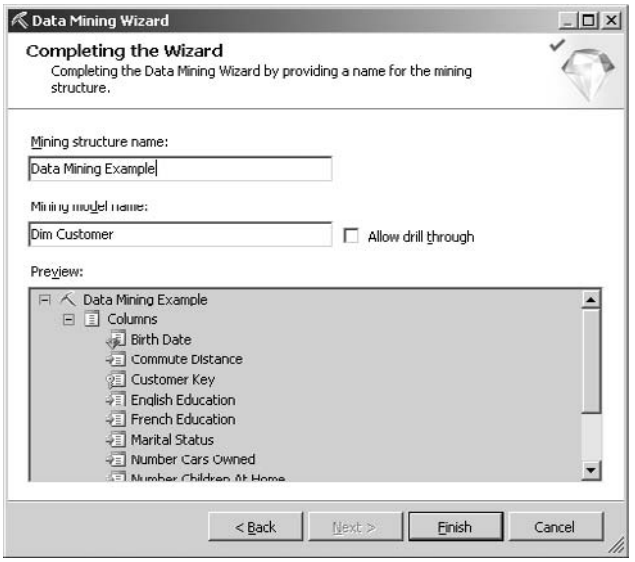


12. Click OK to return to the wizard.

13. Notice that the Input fields are checked for the fields you specified in the Suggest Related Columns screen.



14. Click Next to continue.
15. On the Specify Columns' Content And Data Type screen, click Next.
16. On the Completing The Wizard screen, enter **Data Mining Example** in the Mining Structure Name box and click Finish.



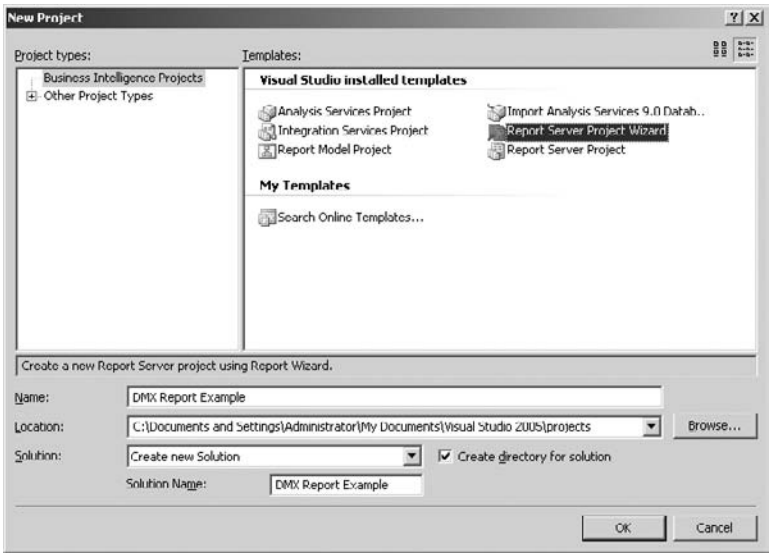
17. In the Solution Explorer, under Mining Models, right-click the Data Mining Example and click Process. If you are asked to build and deploy the project first, click Yes.
18. In the Process Mining Structure dialog box, click Run.

Creating a Report Based on a Data-Mining Model

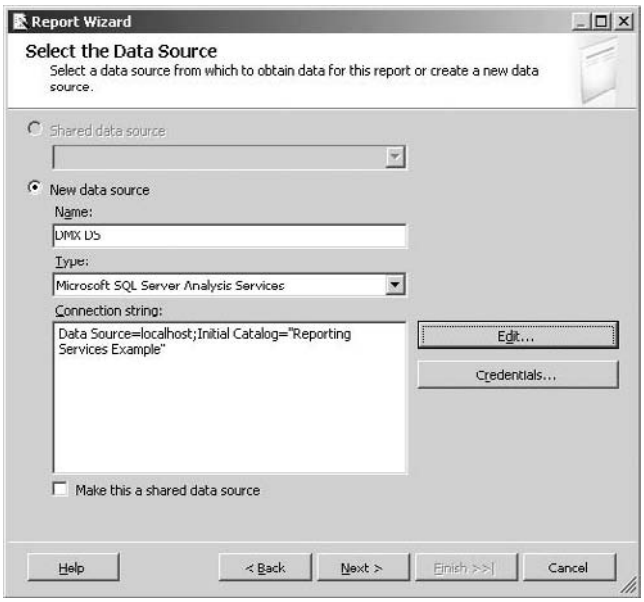
With a new data-mining model in place and trained, you can create a new report based on that model. Let's do that now using the Report Wizard:

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio to open BIDS.
2. Select File ➢ New ➢ Project.

3. Select Business Intelligence Projects from the Project Types list.
4. Select Report Server Project Wizard from the Templates list.
5. Type **DMX Report Example** in the Name text box (note that the Solution Name box is filled in for you).



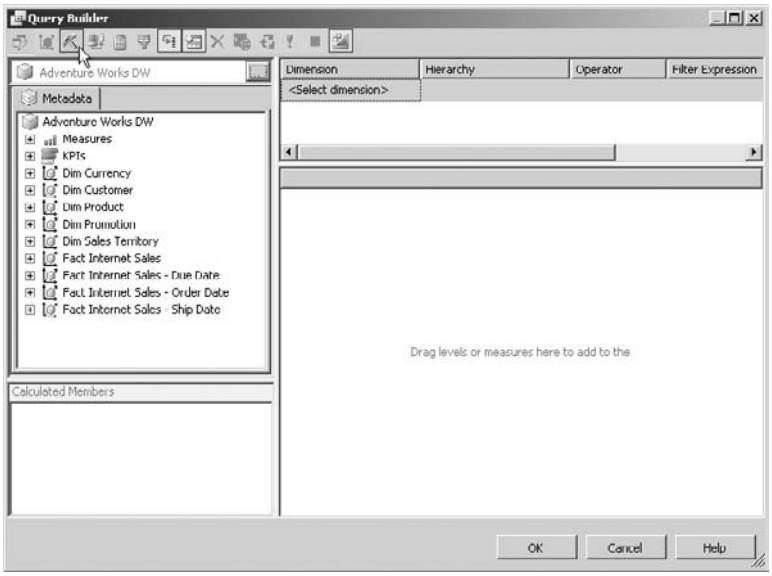
6. Click OK to start the wizard.
7. On the Welcome screen, click Next.
8. On the Select The Data Source screen, type **DMX DS** in the Name box.
9. Select Microsoft SQL Server Analysis Services from the Type drop-down list box.
10. Click the Edit button to create a connection string.
11. Type **localhost** in the Name box (or the name of the server Analysis Services is installed on if it is not your local server).
12. Select Reporting Services Example from the drop-down list box under Select or enter a database name.
13. Click OK to return to the wizard.



14. Click Next to continue.

15. Click the Query Builder button at the top of the Design The Query screen.

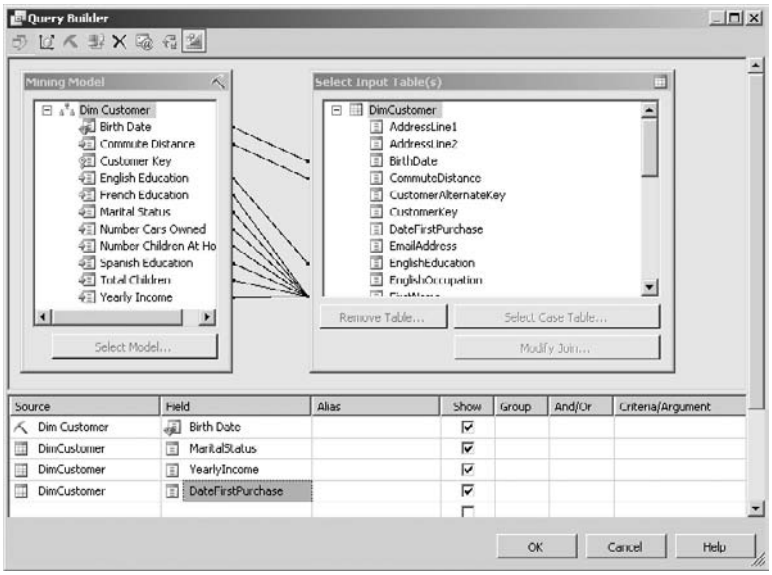
16. On the toolbar at the top of the Query Builder, click the Command type DMX button (it is at the left with an icon that looks like a pickax), then click Yes in the subsequent dialog box.



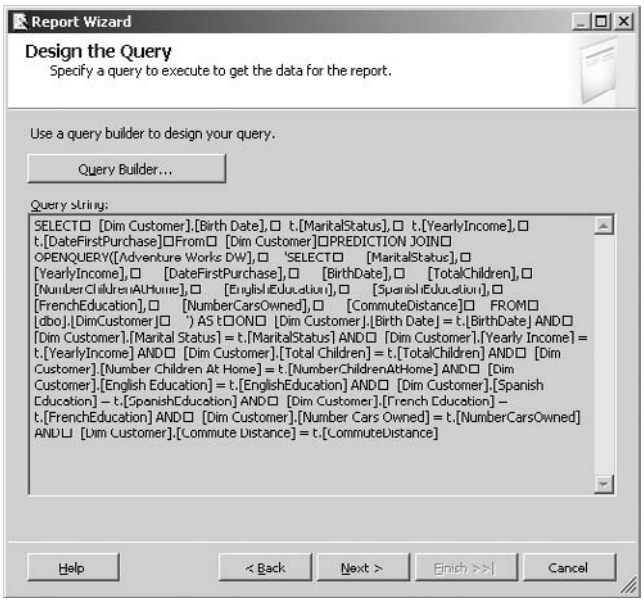
17. In the Mining Model box, click the Select Model button.

18. In the Select Mining Model dialog box, expand Data Mining Example, select Dim Customer, and click OK.

19. In the Select Input Table(s) box, click the Select Case Table button.
20. Select DimCustomer and click OK.
21. Under Source in the bottom pane, select Dim Customer Mining from the drop-down list.
22. Under Field, select Birth Date from the drop-down list.
23. In the next row, select DimCustomer Table from the source list and select MaritalStatus from the field list.
24. In the next row, select DimCustomer Table from the source list and select YearlyIncome from the field list.
25. In the next row, select DimCustomer Table from the source list and select DateFirstPurchase from the field list.



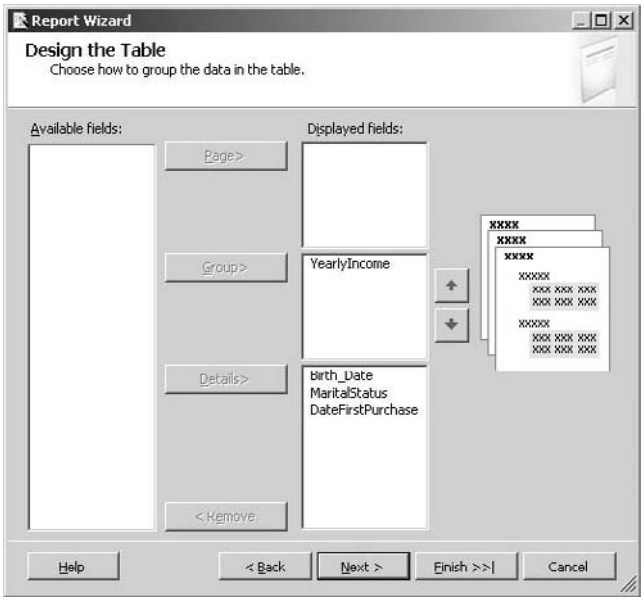
26. Click OK to return to the wizard.



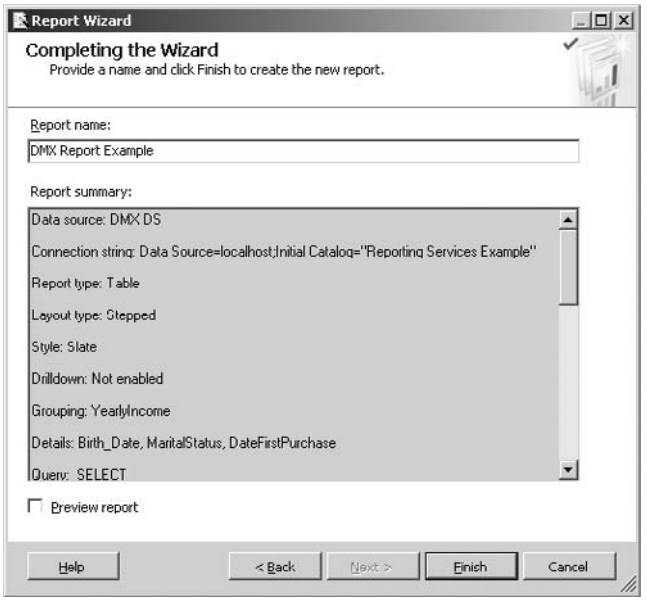
27. Select the Tabular option on the Select Report Type screen and click Next.

28. On the Design The Table screen, select the YearlyIncome field from the Available Fields list and click the Group button.

29. Add the remaining fields to the Details area by selecting them one at a time and clicking the Details button.



30. Click Next to continue.
31. Accept the defaults on the Choose The Table Layout screen and click Next.
32. Select a style on the Choose The Table Style screen and click Next.
33. Accept the defaults on the Choose The Deployment Location screen and click Next.
34. Type **DMX Report Example** in the Report Name box and click Finish to create the report.



Again, there's not much to creating the actual report. The brunt of the effort in creating a data-mining report is spent on actually designing the model and crafting the query to mine the model. This simple example just shows you the marital status, birth date, and date of first purchase for the customer base grouped by their yearly income. In the real world, data-mining reports are much more complex and take a great deal more time to create.

As with an Analysis Services MDX-based report, the biggest difference between a DMX- and a TSQL-based report is the data pane. The data pane in a data-mining report is based on the DMX query designer that you would see when working with a mining model in BIDS.

The Bottom Line

Gain a better understanding of OLAP. Online analytical processing (OLAP) allows you to create and manage multidimensional structures called cubes that contain data that has been collected from other sources.

Master It What are the two types of tables that make up a cube, and what do they contain?

Gain a better understanding of data mining. OLAP is just one of the tools available for data analysis. SQL Server Analysis Services also provides data-mining tools to enhance the data analysis functions.

Master It Describe data mining and give an example of when it would be useful.

Learn how to base a report on a cube or data-mining model. Once you have created a cube or a mining model, you need to be able to create a report based on these data sources so that you can get the information contained in them to the management staff.

Master It Create a report based on the Reporting Services Example cube that shows the list price, dealer price, and order quantity grouped by product name.

Part 2

Using Reporting Services in Your Infrastructure

After you have designed a Reporting Services infrastructure, you are ready to deploy it and put it to use. In this part, I will show you how to deploy your reporting infrastructure based on the plans you crafted in Part 1.

You will also learn how to start using Reporting Services to get some work done. I assume that you already know the basics of creating and publishing a simple report, so I'm skipping that and heading right into more advanced topics. You'll see what is involved in allowing your users to create reports for themselves using the new Report Builder tool. We'll discuss how to create and manage subscriptions, and I'll show you some creative uses for them. You will also learn how to manage and administer your reporting system.

There are four chapters in this section:

- **Chapter 7: Deploying Reporting Services**
- **Chapter 8: Ad Hoc Reporting**
- **Chapter 9: Subscriptions**
- **Chapter 10: Maintenance and Administration**

Chapter 7

Deploying Reporting Services

The last time you took a trip, did you just get the family into the car and start driving into the wild blue yonder, hoping to run across something fun? Of course not; more than likely you spent a considerable amount of time deciding where to go, what to see when you got there, how much to spend, how long to stay, and so on. In other words, you spent a significant amount of time planning the excursion.

We generally plan any complex operation that we undertake, and the more complex the activity, the more time we put into planning. Installing a software system as complex as SQL Server Reporting Services, then, requires that you spend some time planning the deployment.

What happens if you don't spend time planning the Reporting Services deployment? You will probably be able to install the system just fine, and it may seem to work the way you want it — that is, until you try to expand the system or add a new feature. On the other hand, the system may not work properly from the start and you may end up fixing mistakes that could have been avoided early on. Either way, you need to spend time planning the deployment to avoid making such mistakes.

There are several decisions to make when planning a Reporting Services deployment. First, what type of deployment will this be? You have two options to choose from: standard and scaled out. A standard deployment involves one Report Server with one database server. A scaled-out deployment has a single database server with multiple Report Servers, which may be load-balanced for enhanced performance.

You may also decide to install Reporting Services and SharePoint services on the same machine, running side by side. If that is your decision, then you should plan on making some special adjustments to the system so that both pieces of software will work properly when you are done.

After you have deployed Reporting Services, you may have to configure the system, which takes some forethought as well. You must tell Reporting Services how to use custom authorization cookies, send e-mail, and the like.

All of this can take time that, as a busy professional, you may not have. To save time deploying the software, you may decide to automate parts of the deployment using scripting. This chapter explains when to use scripting in the first place, and what can and cannot be scripted.

In this chapter you will learn:

- What types of deployments are available and how to use them
- How to integrate with SharePoint
- How to configure Reporting Services components after installation
- How to automate deployment tasks

Planning a Reporting Services Deployment

Before you install SQL Server Reporting Services on all the Report Server machines in your enterprise, it is vital that you plan the deployment. If you do not plan properly, then you may end up adding features, making fixes, or reinstalling the software on all the machines in your organization. Think of all the extra work and headaches that would cause.

Two types of deployments are available in Reporting Services: standard and scaled out. Let's look at the standard deployment first.

Standard Deployments

A standard deployment of Reporting Services consists of a single Report Server instance, which uses a local or remote database server. The database server can be either SQL Server 2000 or SQL Server 2005.

The biggest decision you need to make when using a standard deployment is whether to make the database local or remote. To make that decision, you need to know what kind of load the Report Server machine is under and what it can tolerate. Windows System Monitor includes a few counters that can help you determine what the load on the hardware is both before and after you install. Table 7.1 shows what those counters are and what their values should be.

TABLE 7.1: Common Counters and Values in System Monitor

OBJECT	COUNTER	RECOMMENDED VALUE	USE
Processor	% Processor Time	Less than 75%	The amount of time the processor spends working.
Memory	Pages/Sec	Fewer than 5	The number of times per second that data had to be moved from RAM to disk, and vice versa.
Memory	Available Bytes	More than 4 MB	The amount of physical RAM available. This number should be low, because NT uses as much RAM as it can grab for file cache.
Memory	Committed Bytes	Less than physical RAM	The amount of RAM committed to use.
Disk	% Disk Time	Less than 50%	The amount of time that the disk is busy reading or writing.
Network Segment	% Network Utilization	Less than 30%	The amount of network bandwidth being used.

NOTE

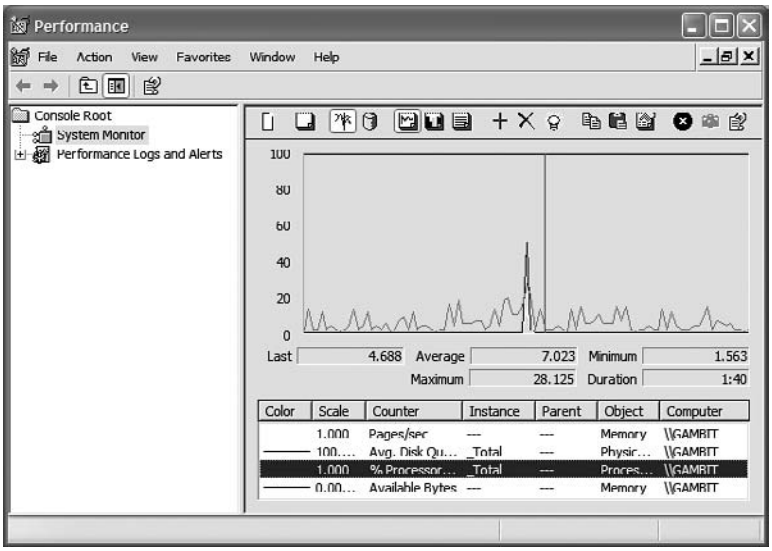
To see the Network Segment: % Network Utilization, you must install the Network Monitor Agent. To access this feature, double-click Network in Control Panel and select the Services tab.

If any of the counters in Table 7.1 are close to or out of the specified range, you should not install the database server and the Report Server on the same machine. If the counters are well below the specified range, you can consider installing both the database server and Report Server on the same machine. You will still need to take disk space into account. First, though, if you have not used System Monitor before, it is a good idea to get some practice with it.

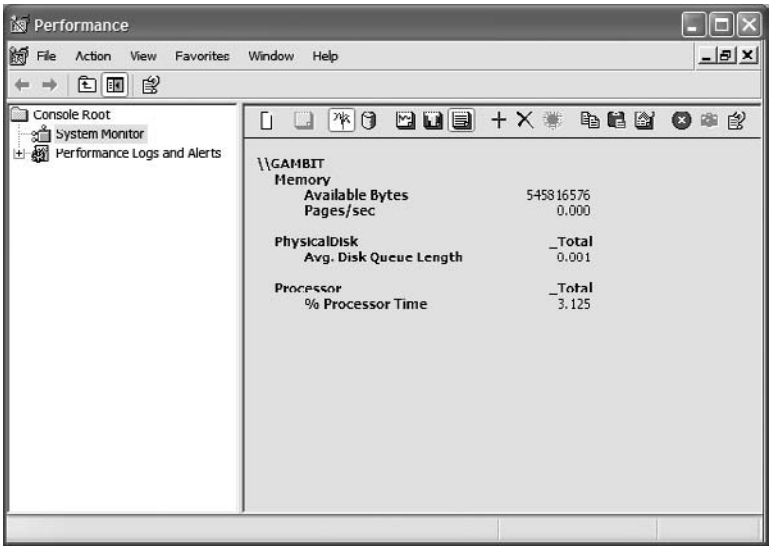
In the following steps, you will add some counters to the Windows System Monitor and work with the chart, report, and alert views:

1. Log into Windows as Administrator.
2. From the Start menu, select Programs ➢ Administrative Tools ➢ Performance. Notice that the graph is already populated with counters.
3. On the toolbar, click the Add button (it looks like a plus sign) to bring up the Add Counters dialog box.

4. From the Performance Object drop-down list, select Memory.
5. In the Select Counters From List box, select Available Bytes and click Add.
6. Click Done and notice the graph being created on the screen.

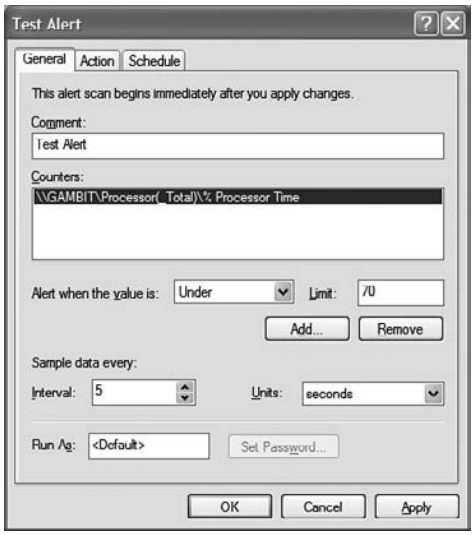


7. Press Ctrl + H and notice that the current counter turns white. This makes the chart easier to read.
8. On the toolbar, click the View Report button (it looks like a sheet of paper) and notice how the same data appears in Report view.



9. In the left pane, expand Performance Logs And Alerts, right-click Alerts, and select New Alert Settings.
10. Enter **Test Alert** in the Name box and click OK.

11. In the Alert Settings dialog box, enter **Test Alert** in the Comment field.
12. Click Add.
13. Select the Processor object and the % Processor Time counter; click Add; and then click Close.
14. Select Under from the Alert When Value Is drop-down list, enter **70** for the Limit, and click OK. This will generate an alert if the processor is not busy 70 percent of the time. In the real world, this would be set to Over 70%, thus warning you just before processor usage becomes a serious problem.



15. To view the alerts, open Event Viewer and look for them in the Application log.

16. Watch the alerts generated for a short time, then select the alert and press the Delete key on your keyboard.

17. Exit System Monitor.

The other issue to consider when using a standard deployment is disk space. The Report Server database takes up space on the hard drive, so you need to make sure there is enough space. Don't be fooled by the initial database size either. The database is small when it is first installed, but it grows quickly depending on the number of reports and the number of users viewing them. Several things add to the overall size of the database:

Reports, Folders, Shared Data Source Items, and Metadata Report definitions, folders, shared data sources, and metadata like schedules, subscriptions, and properties are stored in the Report Server database. These items do not take as much space as the other items listed, but they are worthy of consideration.

Resources A resource is an object that is stored along with your report. It is usually an image file or a collateral document. Resources can take up quite a bit of space in the database depending on the size of the file. These are stored as binary large objects (BLOBs) in the database.

Session State Information Reporting Services stores session state information in the Report Server temporary database. For each session created, Reporting Services creates a row in the table. So, unless you have a significant number of users, this data is not going to take an excessive amount of space on your disk.

Cached Reports, and Report History and Execution Snapshots Cached reports are stored in tables in the Report Server temporary database. These can be quite large, because they contain the actual query results that the report is based on. If the report accepts parameters, there will be a cached version for every combination of parameters for the report, so you should allocate considerable space for these reports.

Snapshots are stored in the Report Server database and not in temporary tables. Like cached reports, snapshots may contain a sizable rowset. If you are using report history to archive reports, you need to allocate disk space accordingly.

Unless you have budget or equipment constraints, it is wise to install the Report Server and the database engine on separate machines for the best possible performance. Figure 7.1 illustrates this concept.

Scaled-Out Deployments

As Figure 7.2 shows, a scaled-out deployment is composed of two or more Report Servers, referred to as *nodes*, which are connected to the same database. This can only be done in production with Enterprise Edition.

> **NOTE**
>
> **You can also deploy a scaled-out implementation using Developer and Evaluation Editions, but these are not licensed for production use.**

You can cluster your database servers for added reliability, but you cannot cluster the Report Servers in the web farm. You can, however, load-balance them to improve performance. You can

load-balance the Report Servers using the built-in Windows Load Balancing software or by using a third-party hardware or software implementation.

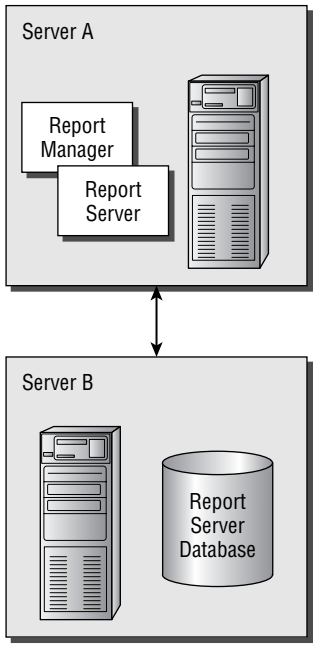


FIGURE 7.1
A standard deployment involves one Report Server and one database server, which can be local or remote.

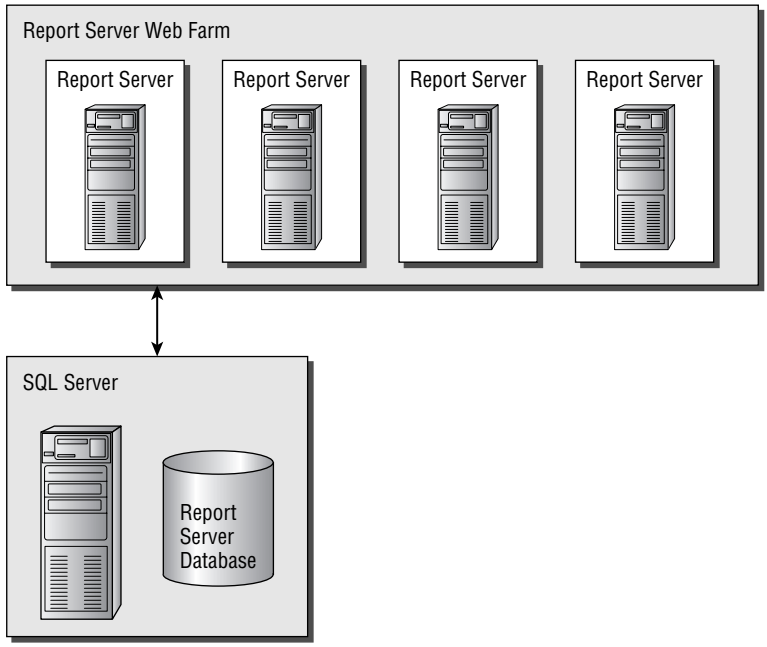


FIGURE 7.2
A scaled-out deployment involves more than one Report Server connected to a single database server.

Here are a few other points to consider before you decide to implement a scale-out deployment as well:

- All of the Report Servers must be the same version and service pack level.
- All of the servers involved must be in the same domain or a trusted domain.
- If you are running Service Pack 2, all of the servers must be in the same mode of operation: native or SharePoint integrated (more on that later in this chapter).

Installing Reporting Services on multiple servers in a web farm is similar to a standard deployment. In fact, you install Reporting Services on the first server in the web farm using the same methods you use to install a standard deployment. On subsequent servers in the web farm, the procedure changes slightly. Let's look at the steps for the whole process.

On the first server in the web farm, follow these steps to install Reporting Services:

1. Install Reporting Services on the first Report Server.
2. Once that is done, start Reporting Services Configuration, and then configure the virtual directories and service accounts using the Report Server Virtual Directory page, Report Manager Virtual Directory page, and Report Manager Web Service Identity page.
3. On the Database Setup page, type or select the name of the SQL Server instance that will host the Report Server database.

NOTE

Some versions of SQL Server 2005 do not allow remote connections by default, so you might need to enable remote connections on the database server if you are not connecting to a local instance.

4. Click Connect to connect to the SQL Server instance.
5. Click New to create the Report Server database. The SQL Server Connection dialog box appears.
6. Specify an account that has permission to create a database.
7. Click OK to create the database.
8. On the Database Setup page, in Credential Type, select the type of account and credentials that the Report Server will use to connect to the Report Server database.
9. Click Apply. The Reporting Services Configuration tool creates the database and configures the connection.
10. On the Encryption Keys page, back up a copy of the symmetric key. As a best practice, always back up the key when you first create the database.
11. On the Initialization page, verify that an entry for the Report Server instance is created. The Initialized check box should be selected.

On the remaining servers in the web farm, follow these steps to install Reporting Services:

1. Run Setup on the next server in the farm to install another instance of Reporting Services.
2. Use the Reporting Services Configuration tool to connect to and configure the new instance you just installed.

3. Configure the service accounts and virtual directories for the new server instance. These can be the same accounts and virtual directory names that you specified for the first instance, or they can be different.
4. On the Database Setup page, in Server Name, specify the SQL Server Database Engine instance that hosts the Report Server database you want to use. This must be the same server that you connected to in step 3 of the previous set of the instructions.
5. Click Connect.
6. On the Database Setup page, in Database Name, select the database you created in step 5 of the previous set of the instructions. If the database list is empty, repeat the previous two steps to establish a connection to the server.
7. On the Database Setup page, in Credential Type, select the type of account and credentials that the Report Server will use to connect to the Report Server database. These do not have to be the same credentials that you used when creating the first Report Server instance, but they can be.
8. Click Apply. The SQL Server Connection dialog box appears. Click OK to connect to the server. The Report Server connects to the Report Server database and adds its key information, which is required for initialization.
9. On the Initialization page, verify that an entry for the second Report Server instance is created. The second Report Server instance has not been initialized yet, so the Initialized check box should be empty.
10. Initializing a new Report Server instance can only be done from a Report Server that is already initialized, so reconnect to the first Report Server instance. Because it is already initialized for reversible encryption operations, it will be used to initialize the new Report Server instance for scaled-out deployment.
11. On the Initialization page, select the Report Server instance you are joining to the deployment, and click Initialize. After you click Initialize, the Initialized check box will be selected, indicating that the Report Server instance is part of the same scaled-out deployment. You might need to refresh the page to see the change. If the Report Server instance is not listed on the Initialization page, verify that you configured the instance to use the shared database.

You must take these steps to complete the scaled-out deployment:

1. Run the SQL Server Surface Area Configuration tool to verify that Reporting Services is enabled and that the Report Server Windows service is enabled and running.
2. After all the nodes in the deployment are installed and initialized and before you edit the Reporting Services configuration files, you must configure view state validation for Report Manager for every Report Server instance in the deployment. You'll learn more about this later in this section.
3. Be sure to back up all the configuration files that will be modified in subsequent steps so that you can roll back if you make a mistake. These files include:
   - `RSReportServer.config` (in the *`X:`*`\Program Files\Microsoft SQL Server\MSSQL.`*`N`*`\Reporting Services\ReportServer` directory)

- RSWebApplication.config (in the *X:*\Program Files\Microsoft SQL Server\MSSQL.*N**Reporting Services*\ReportManager directory)
- The Web.config files for both the Report Server (in the *X:*\Program Files\Microsoft SQL Server\MSSQL.*N*\Reporting Services\ReportServer directory) and Report Manager (in the *X:*\Program Files\Microsoft SQL Server\MSSQL.*N*\Reporting Services\ReportManager directory).

4. If you have load-balanced the Report Servers, follow these additional steps:
   A. Edit the Reporting Services configuration files to specify the virtual server name (more on this later in this section).

   B. In IIS, set the options on the Home Directory tab of the website's Properties dialog box to configure redirection to the URL of the virtual server.

   C. Verify that you can access the scaled-out deployment through the virtual server name (for example, https://*server*/reportserver and https://*server*/reports). You can check which node processes reports by looking at the Report Server log files or by checking the Reporting Services execution log (the execution log table contains a column called InstanceName that will show which instance processed a particular request).

In step 2 of the last set of steps, I mentioned that you need to configure the view state validation for each of the servers in the web farm. You must do this because view state validation is enabled by default in .NET 2.0, and it uses the ASP.NET process identity to perform validation. The problem here is that validation by process identity does not work in a load-balanced scenario because the process key is different on each machine.

An alternative method is for you to generate a validation key manually and configure each of the servers to use the new key. There are a number of ways to create new keys, such as the autogenerate function, which creates a key for you. An in-depth discussion of the autogenerate function is out of the scope of this book, but the function is described in the .NET Framework documentation. Here are the steps to configure the view state validation:

1. Generate a validation key using the autogenerate function provided by the .NET Framework or by any other means you are comfortable with. Do not use the AutoGenerate option when setting the validationKey attribute.
2. Generate a decryption key. For the decryption key, you can create an explicit value or set decryptionKey to AutoGenerate.
3. Open the Web.config file for Report Manager and set the <machineKey> element. You must specify the validation key, decryption key, and the type of encryption used for validation of data. The following example illustrates the entries you must provide (do not use these examples in your configuration files; they won't work):

```
<machineKey
   validationKey="26ad7rs2a1v3vbe816d44dc22a014f0v3a78s"
    decryptionKey="41573f86f0a8fs1" validation="SHA1"/>
```

4. Repeat these steps for each Report Server in the scaled-out deployment. Verify that all Web.config files in the \Reporting Services\Report Manager folders contain identical <machineKey> elements in the <system.web> section.

I also mentioned in step 4 that you will have to modify the Report Server configuration files to point to the correct virtual server name if you are load-balancing your servers. Here is how to do that:

1. Open the `RSWebApplication.config` file in a text editor.
2. Set `<ReportServerUrl>` to the correct virtual server name and remove the entry for `<ReportServerVirtualDirectory>`. This step ensures that all requests coming through Report Manager are load-balanced to the Report Servers that are running in the scaled-out deployment. The following example illustrates the syntax you should use; it specifies the virtual server name and Report Server virtual directory:

```
<ReportServerUrl>https://MyReportPortal➢
/reportserver</ReportServerUrl>➢
<ReportServerVirtualDirectory>➢
</ReportServerVirtualDirectory>
```

3. Open the `RSReportServer.config` file.
4. Set `<UrlRoot>` to the virtual server address. This step ensures that all hyperlinks in reports point back to the scaled-out deployment and are load-balanced accordingly. This setting is also used to complete report delivery. The following example illustrates the syntax you should use:

```
<ReportServerUrl>https://MyReportPortal➢
/reportserver</ReportServerUrl>
```

5. Repeat these steps for each Report Server in the scaled-out deployment.

Real World Scenario

CASE STUDY: TALKSALOT COMMUNICATIONS

When Talksalot first started using Reporting Services, they were a small company, so they did not need a farm of servers providing reporting capabilities. We decided to implement a standard deployment that served them well for some time.

As time went by, Talksalot grew and their reporting needs grew with them. Fortunately, we had installed SQL Server 2005 Enterprise Edition in preparation for such an eventuality so we were able to deploy other servers in a scaled-out deployment.

We clustered the original database server to make sure that it would always be available, and we load-balanced the Report Servers so that they would respond to client requests at maximum speed.

Another common configuration that you may deploy is a side-by-side implementation with SharePoint.

Playing Nice with SharePoint

SharePoint is a suite of collaboration software from Microsoft that is composed of two components. The first is Windows SharePoint Services (WSS), which administrators can use to create

collaboration websites that allow users to share documents, join discussion groups, fill out surveys, and so forth. WSS is a free download from Microsoft.

The second component is Office SharePoint Server 2007. This enhances WSS by providing more advanced features such as personalization and advanced search functionality. This is not a free download, though a trial version is available.

The main reason that people use software like SharePoint is because they want to share information. Reporting Services is designed for the sole purpose of sharing information, so it makes sense that you may need the two to work together on the same machine. There are two ways to accomplish this goal.

The first method is called a side-by-side installation. In this type of installation, Reporting Services is configured to run on the same machine as SharePoint, but the two are configured to run as separate applications with little interaction.

The second method is a little more advanced. Prior to SQL Server 2005 Service Pack 2, only one mode of operation was available for deployment. With SQL Server 2005 Service Pack 2, Reporting Services can be configured to run in one of two modes: native or SharePoint integrated. So, in the second method, you can configure Reporting Services to run in SharePoint integrated mode, which allows users to save reports in and access reports directly from SharePoint document libraries. In fact, much of Reporting Services' functionality is replaced by SharePoint functionality in integrated mode.

We will look at the nuances of deploying in both of these methods.

NOTE

At the time of this writing, SQL Server 2005 Service Pack 2 is still in beta and may change before final release.

Side-by-Side Installations

In a side-by-side installation, you have SharePoint and Reporting Services running on the same server but they are completely separate from one another. SharePoint does not provide any services to Reporting Services, and vice versa.

When would you want to use this method? Suppose that you already have a web server running SharePoint for your corporate intranet and you do not have a budget for more hardware. Also, you want to provide reports to users that should not have any access to the SharePoint site. A side-by-side installation would allow you to install Reporting Services while still running SharePoint without purchasing more hardware or allowing extra users access to the SharePoint site.

If you decide to use a side-by-side installation, you should install SharePoint first, then Reporting Services. This is because the SharePoint configuration files must be modified to work properly with Reporting Services. If SharePoint is installed before Reporting Services, the Reporting Services Configuration tool will automatically detect the SharePoint installation and make the needed configuration changes.

If you have to install Reporting Services first for some reason, you will need to make the SharePoint configuration changes manually using the SharePoint Administration command-line tool, STSADM, which you can find in `C:\Program Files\Common Files\Microsoft Shared\Web server extensions\12\bin`. Run these two commands:

```
STSADM.EXE -o addpath -url➢
http://localhost/ReportServer -type exclusion
STSADM.EXE -o addpath -url ➢
http://localhost/Reports -type exclusion
```

You should also make sure that SharePoint and Reporting Services are running under different application pools in Internet Information Services (IIS). The Reporting Services Configuration tool will help you create a new application pool for Reporting Services.

SharePoint Integrated Mode

In SharePoint Integrated mode, SharePoint 3.0 provides the storage and security functionality for Reporting Services. This is useful if you have a SharePoint site installed and you want to provide Reporting Services access to everyone who has SharePoint access. Using an integrated deployment in this scenario will cut your administrative duties in half because you will need to configure security and storage settings in only one place: SharePoint.

In fact, you have to configure all of these settings in SharePoint because after the Report Server is configured for integrated mode, the Report Manager Virtual Directory page will be disabled.

Because Reporting Services cannot be installed in SharePoint integrated mode, you must install Reporting Services in native mode and then configure it for integration. Take these steps to configure Reporting Services for SharePoint integrated mode:

1. Install SQL Server 2005 Service Pack 2.
2. If the Report Server and SharePoint web application are installed on separate computers, install the SharePoint object model on same machine with the Report Server. We discuss this in greater depth later in this section.
3. Start the Reporting Services Configuration tool and connect to the Report Server instance that you want to configure for SharePoint integration.
4. If this is a new installation, select the default website and enter a name for the virtual directory on the Report Server Virtual Directory page. Skip the page for configuring the Report Manager virtual directory because you cannot use Report Manager with a Report Server that is configured for SharePoint integrated mode.
5. On the Database Setup page, create a new Report Server database in SharePoint integrated mode by selecting the Integrate With SharePoint Products And Technologies option when you create the database. There is more on this later in this section.
6. Back up the encryption key.
7. If you are working with a new installation of Reporting Services, you should also configure the unattended report processing account if you want to support unattended report execution. Optionally, configure the Report Server for e-mail delivery.
8. Close the Reporting Services Configuration tool.

In step 2 of this set of instructions, I mentioned that you will need to install the SharePoint object model on the Report Server if the two are installed on different machines. This is because Reporting Services relies on SharePoint for security and storage in integrated mode, so Reporting Services needs to know the hierarchy to function. To install and configure the SharePoint object model, follow these steps:

1. On the server that runs Reporting Services, run Setup for Windows SharePoint Services 3.0.
2. On the Installation Types screen, click Advanced.
3. On the Server Type screen, select Web Front End.
4. Click Install Now.

5. After the installation is complete, run the SharePoint Products And Technologies Configuration Wizard to join the Web front end to the remote SharePoint farm.
6. In the Connect To A Server Farm page, select Yes to connect to an existing server farm. Then click Next.
7. On the Specify Configuration Database Settings page, specify the SQL Server Database Engine instance that hosts the configuration database.
8. Enter a username and password that the web front end will use to connect to the configuration database, and then click Next.
9. On the Completing The SharePoint Products And Technologies Wizard page, click Advanced Settings.
10. In the Advanced Settings page, verify that the Do Not Use This Machine To Host The Website option is selected.
11. Click OK, click Next, and then click Finish to complete the wizard.

I also mentioned in step 5 of the earlier series of steps that you would need to create a Report Server database in SharePoint integrated mode. When configured for SharePoint integration, the Report Server database stores a primary copy of the following:

- Server properties
- Report execution snapshots
- Report history
- Subscription definitions
- Schedules

Also, to improve performance, the Report Server database stores a secondary copy of:

- Reports
- Report models
- Shared data sources
- Resources Configuring Reporting Services Components

Here are the steps to create a SharePoint integrated database:

1. Click Start ➢ Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ Reporting Services Configuration and connect to the Report Server instance you want to configure.
2. Click Database Setup to open the Database Setup page.
3. Enter the name of the SQL Server Database Engine you want to use.
4. Click Connect.
5. Click New.
6. In the SQL Server Connection dialog box, enter a name for the new database.
7. Select the Create The Report Server Database In SharePoint Integrated Mode check box.

8. Click OK.
9. On the Database Setup page, specify the credentials that are used to connect to the Report Server database:
   - Choose Service Credentials to use the Windows service account and Web Service Account to connect through integrated security.
   - Choose Windows Credentials to specify a domain user account specified as domain_nameuser_name.
   - Choose SQL Server Credentials to specify a SQL Server login.
10. Click Apply.
11. Restart the Report Server Windows service.

Automating Deployment

Installing Reporting Services on a large number of servers throughout the organization can be a daunting task. To speed up the process, Reporting Services supports scripting functionality to automate some of the routine installation and configuration tasks.

Not every step in the process should be automated. For example, if you need to create the same set of reports on every Report Server during the configuration process, it is easier to copy the Report Server database to the new server rather than try to write code to create the new reports automatically.

Some of the installation and configuration tasks will require custom code to automate. For instance, you can automate the configuration of the virtual directories in IIS using custom code that calls the Report Server Windows Management Instrumentation (WMI) provider. If you do not want to write custom code, then you will have to configure these tasks manually. Let's take a look.

Automating the Installation and Configuration Steps

In this section we'll look at the recommended approaches.

Install Reporting Services

The Reporting Services setup program provides a command-line interface that you can use to run a silent installation. The command looks something like this:

```
Setup /settings c:\customsetup.ini /qn
```

This requires you to create a special initialization, or INI, file for the setup program. A sample INI file looks like this:

```
[Options]
INSTANCENAME=MSSQLSERVER
   ADDLOCAL=SQL_Engine, SQL_Data_Files, RS_Server,
RS_Web_Interface,SQL_WarehouseDevWorkbench,
SQL_ComputerManagement,SQL_Workbench,RS_tools
RSACCOUNT=domain\account
RSPASSWORD=password
```

```
RSUPGRADEDATABASEACCOUNT=sql_login
RSUPGRADEPASSWORD=password
RSAUTOSTART=1
RSCONFIGURATION=Default
RSSQLLOCAL=1
```

Here is what each setting is used for:

INSTANCENAME This is the instance of the database engine you are connecting to. The default is MSSQLSERVER. If you are connecting to an instance other than the default, change this setting to match.

RSACCOUNT This specifies the account under which the Report Server Windows service runs. This setting is optional.

RSPASSWORD This is the password for the account specified by the RSACCOUNT property. This setting is optional.

RSUPGRADEDATABASEACCOUNT Specifies a SQL Server database login used to connect to the SQL Server instance that hosts the Report Server database during an upgrade. This setting is optional.

RSUPGRADEPASSWORD This property specifies the password for the user account specified in the RSUPGRADEDATABASEACCOUNT property. This setting is required if the RSUPGRADEDATABASEACCOUNT property is set and an empty string is a valid value.

RSCONFIGURATION Specifies how a Report Server instance is installed. It is optional. There are two settings for this property. The default installation requires a local Database Engine instance and a system configuration that uses default values. The Database Engine instance cannot be installed as a virtual server.

The files-only installation installs the program files and minimally configures a Report Server installation. This is the default value if this property is not configured in the initialization file. You will need to perform further configuration after installation is complete.

RSAUTOSTART If this is set to true (the default), then the Report Server Windows service is started automatically when Windows is restarted. If it is set to false, then the service does not automatically start.

RSSQLLOCAL Setting this to 1 (the default) indicates that the SQL Server database engine is installed on the same server as the Report Server. A setting of 0 indicates that the database engine is on a remote server.

ADDLOCAL This specifies software components that will be installed locally. Some of the options are:

- SQL_Engine: Installs the SQL Server 2005 database engine.
- SQL_Data_Files: Installs the SQL Server 2005 databases (Master, MSDB, Model, Tempdb).
- RS_Server: Installs the Reporting Services engine.
- RS_Web_Interface: Specifies that Report Manager will be installed.
- SQL_WarehouseDevWorkbench: Specifies that Business Intelligence Development Studio will be installed.

- `SQL_ComputerManagement`: Installs the SQL Server 2005 Computer Manager application.
- `SQL_Workbench`: Installs Management Studio (which was called Workbench during the SQL Server 2005 beta period).
- `RS_tools`: Installs the Reporting Services command-line tools.

Configure a Website in IIS

The Reporting Services setup program does not provide a command-line method to create and configure a website, but IIS 6.0 does. You can create a new website using the `iisweb.vbs` script, which is stored in the *`systemroot`*`\system32` folder. The syntax for the command is:

```
iisweb /create Path SiteName /b Port /i IPAddress /d HostHeader /dontstart /s
Computer /u Domain\User /p Password
```

Here is what each parameter is used for:

`Path` Specifies the location of content files for the root directory of the website. This must be a local path, such as `C:\Intepub\wwwroot\HTML`; `iisweb.vbs` will create the path if it does not exist. The `Path` parameter must immediately precede the `SiteName` parameter in the command or `iisweb.vbs` will not interpret the site information correctly.

`SiteName` Specifies a name for the website. This setting is required.

`/b Port` Specifies a TCP port number for the website. The default port is 80.

`/i IPAddress` Specifies an IP address for the website. The default is All Unassigned, which assigns to the site all the IP addresses on the computer that are not assigned to other sites. Only one site on each IIS server can be set to All Unassigned.

`/d HostHeader` Specifies the host header name for the website, such as `www.sybex.com`. By default, the site does not have a host header name and must be identified by its IP address or port number.

`/dontstart` Websites are automatically started when IIS is started so that they can service requests. This parameter tells IIS not to start the site when IIS is started so that it will not service requests.

`/s Computer` By default, `iisweb.vbs` runs on the local system. This parameter runs the script on the specified remote computer. Do not use backslashes (\) with this parameter; just type the computer name or IP address by itself.

`/u Domain\User` Runs the script with the permissions of the specified user account, which must be a member of the Administrators group on the remote computer.

`/p Password` Specifies the password of the user account that is specified in the `/u` parameter.

The following sample command creates a website on the local system named Reports using the IP address of 192.168.200.21 pointing to the `c:\inetpub\reports` directory:

```
iisweb.vbs /create c:\inetpub\reports "Reports" /i 192.168.200.21
```

Configure the Report Server Web Service and Virtual Directories

There are no command-line utilities available to automate this process, so you will need to write custom code that accesses the WMI provider for Reporting Services. This approach is only useful if you have a large number of Report Servers to install.

Create, Upgrade, or Configure the Report Server Database

To automate this task, you can create the `GenerateNewDatabaseScript.sql`, `UpgradeDatabaseScript.sql`, or `GrantRightsScript.sql` file from predefined templates or by using the Reporting Services Configuration tool. You can then run these scripts using the `osql.exe` command-line tool.

Configure a Scaled-Out Deployment

You can write custom code that calls the Reporting Services WMI provider to do this, or you can run the `rskeymgmt.exe` utility.

Back Up Encryption Keys

You can also use the `rskeymgmt.exe` utility or write custom code to back up the encryption keys. The command line for the `rskeymgmt.exe` command looks like this:

```
rskeymgmt -e -f x:\rsdbkey.snk -p password
```

Configure Report Server E-mail

You will need to write custom code that calls the WMI provider to automate this task. You may be tempted to copy the `RSReportServer.config` file to automate this task, but do not copy the file. The `RSReportServer.config` file is specific to each machine it is installed on.

Configure the Unattended Execution Account

You can create custom code to call the WMI provider to automate this task or you can use the `rsconfig.exe` utility. The command looks like this:

```
rsconfig -e -u domain/username -p  password
```

Postdeployment Configuration

After you have deployed Reporting Services, you can start using it right away. Some special functions require further configuration before you can make use of them, though. In this section, we'll look at some of the features that require further configuration and what you need to do to make them work.

Custom Authorization Cookies

By default, Reporting Services does not pass custom authorization cookies through to end users; it will only transmit cookies specific to the server Reporting Services is installed on. There are instances, though, when it is necessary to configure Reporting Services to transmit these custom cookies. A good example is if you are using a single sign-on (SSO) application.

SSO applications are used to enhance security by consolidating user logon information for disparate systems into a single repository. Then, users need to sign on to the network only once and the SSO application will provide logon information to the networked systems that require a logon.

NOTE

Microsoft Office SharePoint Server 2007 provides SSO functionality if you are looking for such a tool.

Most SSO applications generate a custom authorization cookie to provide this service. The problem is that Reporting Services does not transmit these custom cookies by default, so you have to make a configuration change.

To enable Reporting Services to retransmit these custom authorization cookies, you need to add the <PassThroughCookies> section to the RSWebApplication.config file, like so:

```
<UI>
   <CustomAuthenticationUI>
   ...
      <PassThroughCookies>
         <PassThroughCookie>
            cookiename1
         </PassThroughCookie>
         <PassThroughCookie>
            cookiename2
         </PassThroughCookie>
      </PassThroughCookies>
   </CustomAuthenticationUI>
   ...
</UI>
```

Advanced E-mail Delivery Settings

Out of the box, Reporting Services comes with an e-mail delivery extension that allows you to distribute reports via e-mail. You can configure the basic settings for e-mail delivery through the Reporting Services Configuration tool, as shown in Figure 7.3. There are some more advanced settings that you can configure by modifying the RSReportServer.config and RSWebApplication.config files.

Let's look at some of the advanced settings you may need to configure.

Setting the To Field

By default, when a user creates a subscription and requests e-mail delivery, the name in the To field of the e-mail is set based on the domain username of the individual who created the subscription. This is a problem if you use e-mail accounts that differ from your domain accounts because the e-mail will not get delivered.

To work around this issue, and get your reports delivered, follow these steps:

1. Open RSReportServer.config with a text editor.
2. Set SendEmailToUserAlias to False.
3. Set DefaultHostName to the Domain Name System (DNS) name or IP address of the SMTP server or forwarder.
4. Save the file.

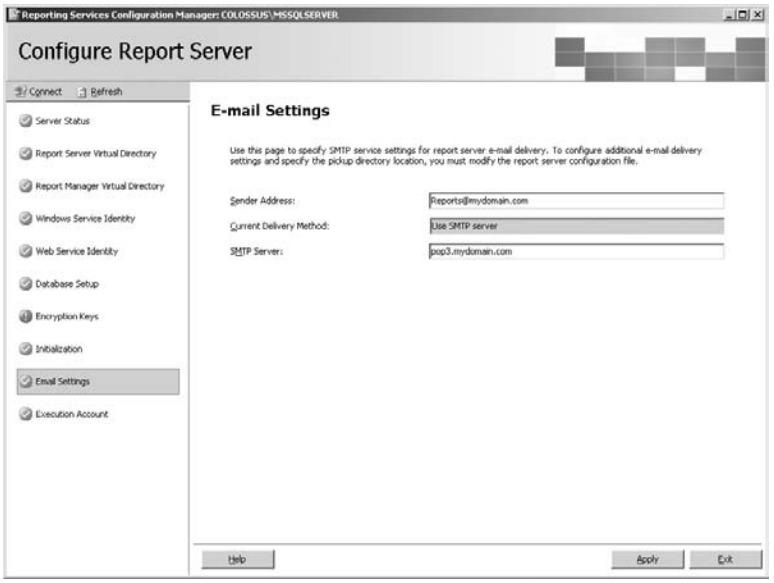


FIGURE 7.3
You can configure basic e-mail settings using the Reporting Services Configuration tool.

The e-mail configuration section of the file would then look something like this:

```
<RSEmailDPConfiguration>
   ...
   <SendEmailToUserAlias>
        True
   </SendEmailToUserAlias>
   <DefaultHostName>
        Pop3.myotherdomain.com
   </DefaultHostName>
   ...
</RSEmailDPConfiguration>
```

SETTING THE SMTP PORT

The default SMTP port is 25, and that is what Reporting Services is configured to use by default. You may have configured your e-mail server to use another port, perhaps for security reasons. If that is the case, then you can change the port that Reporting Services uses for e-mail access by changing the SMTPServerPort setting in the `RSReportServer.config` file. If you are using port 37337, for example, it would look similar to this:

```
<RSEmailDPConfiguration>
   ...
   <SMTPServerPort>
      37337
   </SMTPServerPort>
   ...
</RSEmailDPConfiguration>
```

Setting Authentication

Reporting Services connects to the SMTP server using anonymous authentication, which is the default authentication method for most SMTP applications. You may have some security measures in place that prohibit applications from sending mail anonymously. If that is so, then you can set the `SMTPAuthenticate` setting from 0 (anonymous) to 2, like this:

```
<RSEmailDPConfiguration>
   ...
   <SMTPAuthenticate>
      2
   </SMTPAuthenticate>
   ...
</RSEmailDPConfiguration>
```

Using a Local SMTP Server

You can configure Reporting Services to use a local SMTP server instead of a remote server, which is the default. This option is primarily used for troubleshooting and testing. Here are the steps to configure a local SMTP server and configure Reporting Services to use it:

1. In Control Panel, click Add Or Remove Programs.
2. Click Add/Remove Windows Components to start the Windows Component Wizard.
3. Select Application Server and click Details.
4. Select Internet Information Services (IIS) and click Details.
5. Select the SMTP Service check box and click OK.
6. In the Windows Component Wizard, click Next. Click Finish.
7. Verify that the service is running in the Services console.
8. Open the `RSReportServer.config` file in a text editor.
9. Verify that <`UrlRoot`> is set to the Report Server URL address.
10. Find <`ReportServerEmail`> under Delivery.
11. Clear any settings in the <`SMTPServer`> section, but do not delete the tags.
12. Set <`SendUsing`> to 1.
13. Set <`SMTPServerPickupDirectory`> to a folder on the local drive.
14. Set <`From`> to an account that has permission to send e-mail from the SMTP server.
15. Save the file.

Internet Access

Reporting Services was not designed for use on the Internet, so you need to make some configuration changes if you want to expose it to the Internet.

To start, Report Manager and Report Server are installed as a single unit. On an intranet, this is ideal because it cuts down on install and configuration times. On the Internet, this is bad because

it exposes the Report Server to attack. To make sure that the Report Server is not exposed, you will need to perform a files-only installation on the Internet web server and then run the SQL Server Surface Area Configuration tool to disable the Report Server instance. This will give you a working copy of Report Manager on the Internet web server with a disabled copy of Report Server.

Next, install a copy of Reporting Services on a server behind your firewall, probably on the database server. This will protect Report Server from attack.

After both servers have been installed, configure the Report Manager on the Internet web server to point to the Report Server that is behind the firewall by changing settings in the `RSWebApplication.config` file. Specifically, you need to set the `<ReportServerVirtualDirectory>` and `<ReportServerUrl>` settings to point to the proper URL. For example, if the Report Server is installed on a machine named ExtranetRS in the MyCorp domain and the virtual directory is Reports, the configuration file on the Internet web server would look, in part, like this:

```
<UI>
   ...
   <ReportServerUrl>
       http://ExtranetRS.MyCorp.com
   </ReportServerUrl>
   <ReportServerVirtualDirectory>
       Reports
   </ReportServerVirtualDirectory>
   ...
</UI>
```

A Deployment Checklist

When deploying a complex system, it is easy to miss something along the way. When that happens, it is difficult and time consuming to identify the problem. This handy checklist should help you to avoid some of the more common mistakes when deploying Reporting Services in your environment.

Before you install Reporting Services on any of your systems, check the following:

- Make sure that the system you are hosting the Web server and website on meets the minimum hardware requirements and runs IIS 5.0 or later.
- Verify that the instance of SQL Server that will house the Report Server database has enough storage capacity to handle the database.
- Malicious users can make quick work of damaging an unsecured server, so make sure that your SQL Server is physically secure.
- If you plan to allow e-mail delivery of reports, find the name of your organization's SMTP server.
- Create or identify the accounts that will be used for the web service and Windows service, as well as credentials for connecting to the Report Server database.

After you have completed the setup routine, check these items:

- If you are using Windows Firewall, make sure that port 80 is open. If you installed Reporting Services on Microsoft Windows XP with Service Pack 2 or Windows Server

2003 with Service Pack 1, then port 80 is blocked by default. You can open it by following these steps:

1. Click Start, point to Control Panel, click Windows Firewall, click Exceptions, and then click Add Port.
2. In Name, type **Web Server (TCP 80)** .
3. In Port number, type **80**.
4. For the protocol, select TCP.
5. Click OK.

- If you want added security, you can run the SQL Server Surface Area Configuration tool to make some web and Windows service features unavailable, prevent scheduled operations, and prohibit web service calls.
- If you will be connecting to an Oracle data source, install Oracle client tools on the Report Server. You will need to restart IIS after installing these tools by using the `iisreset.exe` command-line tool.
- The SQL Server Agent service must be running for scheduled operations, like subscriptions, to take place. So make sure that the service is running and that it is set to start automatically when Windows starts.
- You will also need to configure Reporting Services before you can use it. If you installed a default configuration, then this is already done. If you deployed a files-only installation, you must specify virtual directories, accounts, and a Report Server database using the Reporting Services Configuration tool.
- Back up the symmetric key that supports encryption on a Report Server. This can be done using the Reporting Services Configuration tool or the `rskeymgmt.exe` command-line tool.
- Verify that the server is running by opening `http://localhost/reportserver` in a web browser on the Report Server.
- Also, verify that Report Manager is running by opening `http://localhost/reports` in a web browser.
- If you want users to have personal folders for managing and storing individual reports, you can enable My Reports.
- Create role assignments to grant Report Server access to other users. Generally, for each group or user, you should create a pair of role assignments: one for system access and a second one to access items in the Report Server folder namespace. There is more information on this in Chapter 3 of this book.

The Bottom Line

Know what types of deployments are available and how to use them. Two types of deployment are available: standard and scaled-out. Standard deployments involve a single Report Server and a single database server. Scaled-out deployments involve multiple Report Servers using a single database server.

Master It What are the main considerations for implementing a scaled-out deployment?

Know how to integrate with SharePoint. SharePoint is collaboration software that makes it easy for users to work together on projects by sharing documents and graphics, joining discussion groups, and the like. Reporting Services can be configured to work with SharePoint as a separate application on the same machine, or they can be integrated to work together.

> **Master It** Reporting Services cannot be installed in integrated mode, so what steps do you need to take to configure SharePoint integrated mode?

Configure Reporting Services components after installation. Most of the settings that you need to configure to get Reporting Services up and running in production can be made using the Reporting Services Configuration tool. You will need to configure the more advanced settings manually.

> **Master It** What are some of the advanced settings that you need to configure manually?

Automate deployment tasks. When you need to install Reporting Services on a number of servers in your organization, it is usually best to automate the process. Reporting Services supports automation for several steps in the deployment process.

> **Master It** You need to automate the Reporting Services setup program to install Reporting Services, Report Manager, and the Reporting Services command-line tools. You want to use the default configuration, and you need Reporting Services to start automatically when Windows starts. You are installing Reporting Services on the same server as the database engine and you are connecting to the default instance. How would you do this?

Chapter 8

Ad Hoc Reporting

With SQL Server 2000 Reporting Services, when a user needed a new report they had to ask the developers to create and deploy the report. This could be a lengthy process if the developers had a heavy workload. To help the users get reports faster, and lift some of the burden from the developers, Microsoft gave users the ability to create ad hoc reports in SQL Server 2005 Reporting Services.

You will need a grasp of the main components involved in ad hoc reporting before you can deliver the functionality to your users. The first component we'll focus on is Report Builder, the tool users employ to create ad hoc reports. Report Builder is designed to look and feel similar to Microsoft Office products like Excel and PowerPoint, so it is easy to learn.

The reports that users create using Report Builder are not based on databases directly; they are based on report models that developers create. A report model is a descriptive layer of metadata that sits between Report Builder and the database. Why the extra layer?

Database designers and developers generally try to use a descriptive naming convention for their databases and database objects to accelerate development. The problem is that these naming conventions are great for developers, but users cannot make heads or tails of them. For example, suppose you have a table that contains current accounting data for an office in the Pacific Northwest region. The name Acctg_PNW_Curr may make complete sense for a developer but may be gibberish to an end user. Report models allow you to present your end users with objects that make sense to them.

This functionality is not fully configured by default, so you will have to prepare your Report Servers to offer ad hoc reporting functionality.

In this chapter you will learn:

- How to configure your systems to support ad hoc reporting
- How to create and deploy report models
- How to use Report Builder to work with ad hoc reports

Preparing the Systems

Report Builder is a powerful tool that your users can work with to create their own reports. This approach allows users to get the information they need quickly, and it lightens the load for developers. Before your users can start using the tool, you must configure the systems involved, both client and server.

Preparing the Clients

The first step is to prepare the client machines. Report Builder is a `.NET` 2.0 application, so you need to make sure the framework is installed on the client machines. If the framework is not installed

prior to installing Report Builder, your users will be prompted to install it before proceeding. This is not a showstopper, but it can be an annoyance for the users.

It is also important to know that Report Builder is deployed to client machines using ClickOnce technology, which simplifies the deployment process for `.NET` applications. This is important because ClickOnce applications are governed by security policies on the client computer that dictate how the application will run once installed. Report Builder can be installed using full or partial trust. The functionality is the same no matter which security setting you choose, but there are some small differences in the way the application starts up:

- When you choose to install Report Builder using full trust, the end user will get a prompt asking them if they are willing to grant this application full trust permission the first time they run the application. After that, they are not bothered again.
- If you choose to deploy the application using partial trust security, then local Intranet users will not receive a prompt at all and users that are not on the local intranet will receive a prompt asking if they are willing to allow the application to run with restricted permissions. The only drawback is that the users will not be able to access the Report Builder help file in partial trust mode. You will need to provide a copy of the `ReportBuilder.chm` file for them to access manually.

Report Builder is installed using full trust by default. To use partial trust, you need to change the <`ReportBuilderTrustLevel`> setting in the `RSWebApplication.config` file from `FullTrust` to `PartialTrust`. The file will look something like this when you are done:

```
<Configuration>
   <UI>
   ...
      <ReportBuilderTrustLevel>
         PartialTrust
      </ReportBuilderTrustLevel>
   ...
   </UI>
</Configuration>
```

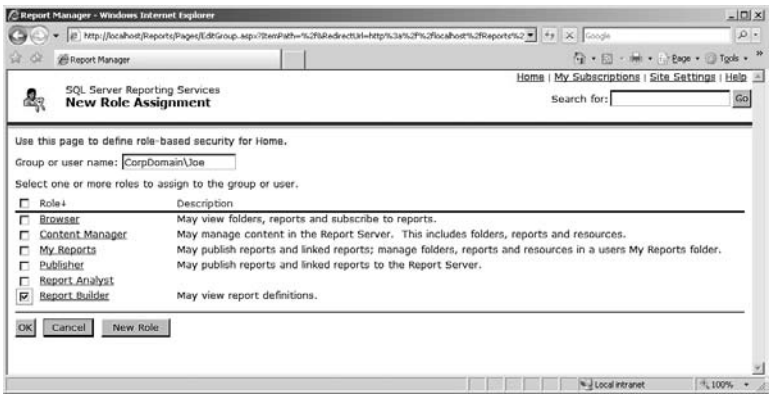
Preparing the Servers

All of the software you need to deploy Report Builder is already sitting on the Report Servers, so you do not need to install any extra software. You do have to configure the security settings to allow access to the application. This process differs slightly depending on whether you are configuring a new system or you have upgraded from a previous version.

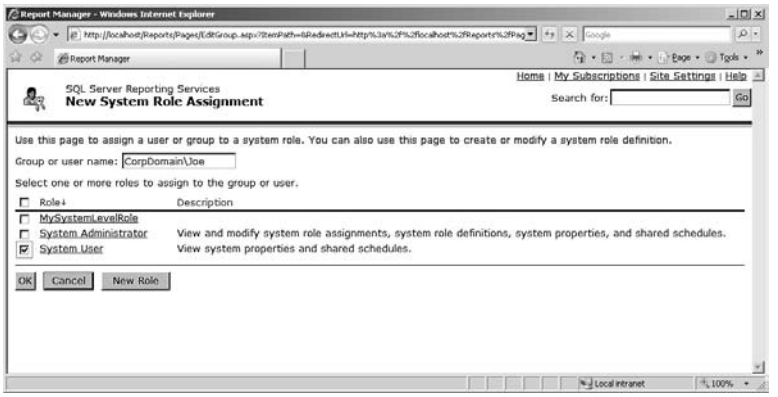
If you are configuring a brand-new system, you can use the predefined roles that were created during setup for this purpose. All you have to do is create a few role assignments for the Report Builder and System User roles. Let's do that now using Report Manager:

1. Open Report Manager and navigate to the Home folder.
2. Select the Properties tab to open the Security page for the Home folder.
3. Click New Role Assignment.
4. In the Group Or User Name text box, type the name of a valid user or group in the ***DomainName\UserName*** format.

5. Check the check box for the Report Builder Role.



6. Click OK to save the new role assignment and return to the Securities page.
7. Click Site Settings.
8. In the Security section, select Configure Sitewide Security to open the System Role Assignments page and click New Role Assignment.
9. In the Group Or User Name text box, type the name of a valid user or group you want to assign a system role to in the ***DomainName\UserName*** format.



10. Check the box next to the System User role.
11. Click OK. The new system role assignments are saved to the Report Server and you are returned to the System Role Assignments page.

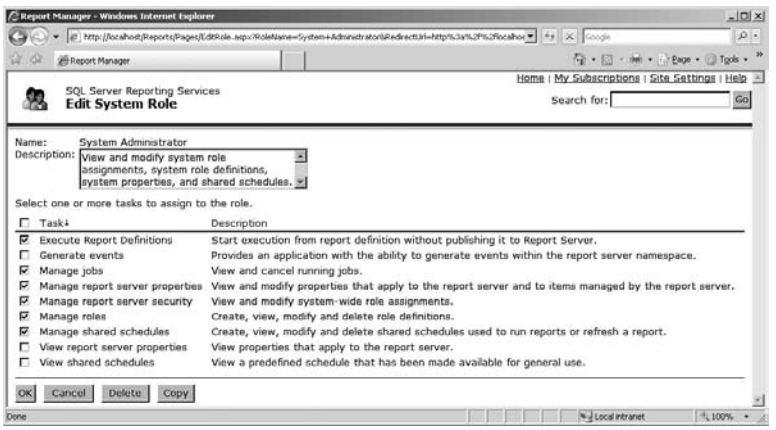
A little more work is involved if you are configuring an upgraded system. Because SQL Server 2005 Reporting Services provides more functionality than its predecessor, it features more tasks to allow access to that functionality. The only problem is that the setup program does not modify existing roles to work with these new tasks, so you will need to update any existing roles manually. Table 8.1 describes the available tasks that affect Report Builder.

TABLE 8.1: Tasks That Affect Report Builder

NAME	LEVEL	DESCRIPTION
View Models	Item	Allows users to view models, use models as data sources, and run queries against models.
Manage Models	Item	Allows users to create, view, delete, and modify models. Also allows users to assign model item security.
Consume Reports	Item	Allows users to edit published model-based reports in their local copy of Report Builder.
Execute Report Definitions	System	Allows users to execute report definitions that are not stored on the Report Server. This is essential because report definitions created with Report Builder are not always stored on the Report Server.
Browse Model	Model Item	Allows a user to view a portion of a model.

You will need to add tasks to the system-level roles to make Report Builder available to the users. Follow these steps to verify that the Execute Report task is assigned to the System Administrator role:

1. Open Report Manager and click Site Settings.
2. Click the Configure System-Level Role Definitions link.
3. Click the System Administrator Role Definition link.
4. Make sure the box next to the Execute Report Definitions task is checked.



5. Click OK to save the changes.

NOTE

See chapter 3 for a detailed discussion of tasks, roles, and role assignments.

When a user opens Report Builder, they make a connection to the Report Server. The type of security you use on the Report Builder folder affects this connection, so you need to choose which authentication method you will use: Windows, Forms, or Basic.

If you decide to use Windows authentication, your users will not be prompted to enter network credentials when they launch Report Builder. Their account information is passed to IIS and they are automatically logged on.

You may also choose to use Forms authentication. When users launch Report Builder with Forms authentication, IIS does not receive their Windows credentials, so they are prompted to log on to the Report Builder site, even if they are already connected to Report Manager.

With Basic authentication, the connections to the Report Builder and Report Server sites are treated separately, so the users will be prompted to enter their credentials twice: once for each service. If you are using Basic authentication on a system that is not part of your domain, you must allow Anonymous access to the Report Builder directory; otherwise, users will not be able to connect. To configure the directory for Anonymous access:

1. Open Internet Information Services Manager in Administrative Tools, expand Web Sites | Default Web Site | ReportServer.
2. Right-click Report Builder and select Properties.
3. On the Directory Security tab, click Edit in the Authentication and Access Control area.
4. Click Enable Anonymous Access, and then click OK.

Working with Report Models

When you create an object in a database, you give it a name that is based on the naming convention that your company has adopted. For example, suppose you work for a company that has been divided into four geographic regions. You have a database for your accounting department named Accounting, and inside that database you have tables that contain accounts payable information for each of the four regions: AcctPay_R1, AcctPay_R2, AcctPay_R3, and AcctPay_R4.

Your users will be able to tell what is in the Accounting database, but they may not know what should be in the AcctPay_R1 table. Even if they do happen to know what is in the table, they may not be able to decipher the column names and data contained within.

To avoid the confusion that this can cause, you need to create a report model. A model is a layer of metadata that describes the database and its relationships in user-friendly terms. This makes the task of creating ad hoc reports in Report Builder much easier for your end users.

Report models can be based on:

- SQL Server 2000 databases
- SQL Server 2005 databases
- SQL Server 2005 Analysis Services Cubes
- Oracle 9.2.0.3 or later databases

You can create models using several tools:

- Model Designer
- Report Manager
- Management Studio
- SharePoint Server 2007

Report models are made of three primary parts: a semantic model, a physical model, and a mapping. A semantic model contains the user-friendly descriptions of objects in the database, and the physical model contains the actual names of the objects in the database. The mapping links the semantic model to the physical model, thus pairing user-friendly object names to their actual names in the database. Figure 8.1 shows how the mapping ties the semantic and physical models.

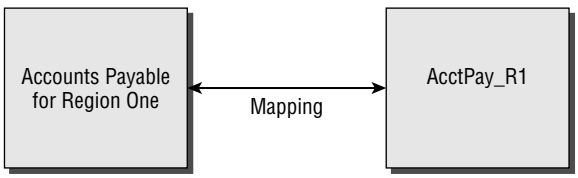


FIGURE 8.1
Semantic models and physical models are tied together by a mapping.

Each report model contains one semantic model, one physical model, and one mapping. Report models are written in the XML-based Semantic Model Definition Language (SMDL) and are stored on disk in an `.smdl` file.

Using Model Designer

The most flexible tool available to create a report model is the Report Model Designer, part of the Business Intelligence Designer Studio (BIDS). You can use this tool to create report models based on SQL Server 2000 and 2005 databases, and Oracle 9.2.0.3 or later databases. You must use other tools to create a model based on an analysis services cube.

A report model project consists of three main components: data sources, data source views, and report model files.

The data source file contains all the information needed to connect to the source database, including the authentication information, data source name, and connection string. Data sources are stored in a file with the `.ds` extension.

The data source view contains descriptions for the tables in the database that the data source point to, the table contents, and the relationships between those tables. It also allows you to make annotations and attach descriptions to objects that are managed outside of the originating data source. This is the source information for the report model.

The report model file contains the metadata description of the database that the data source view is based on. This file contains the semantic model, physical model, and mapping.

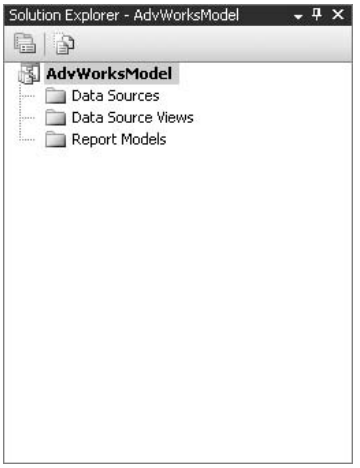
NOTE

For the remainder of this chapter, we are going to assume that you have the AdventureWorks sample database installed on your system.

Creating and Configuring a Project

To start the process, create a report model project in BIDS:

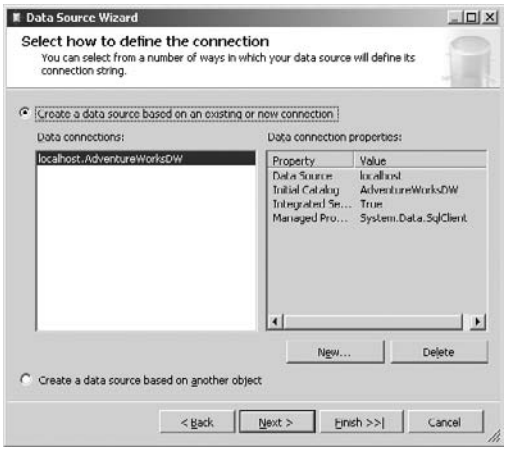
1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio.
2. Choose File ➢ New ➢ Project.
3. Click Report Model Project in the Templates list.
4. In the Name, type **AdvWorksModel**.
5. Click OK to create the model project.



Now that you have a report model project, you need to create a data source to connect to a database. Here, you will create a data source to connect to the AdventureWorks database:

1. In Solution Explorer, right-click Data Sources and click Add New Data Source.
2. On the Welcome screen, click Next.
3. The Select How To Define The Connection page appears, which allows you to create a new data source object or one that is based on an existing data source. You will create a

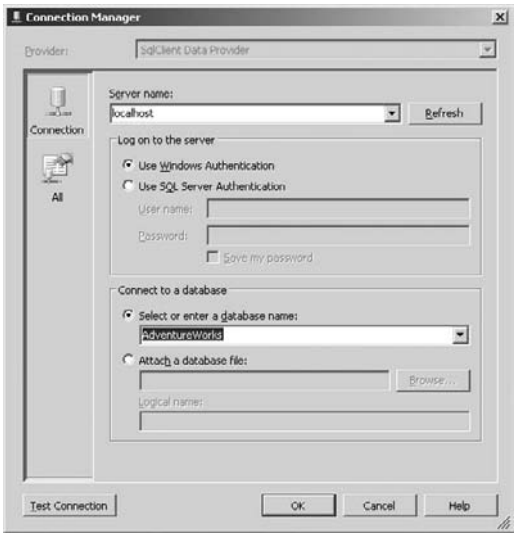
new data source here so verify that ''Create a data source based on an existing or new connection'' is selected, and then click New.



NOTE

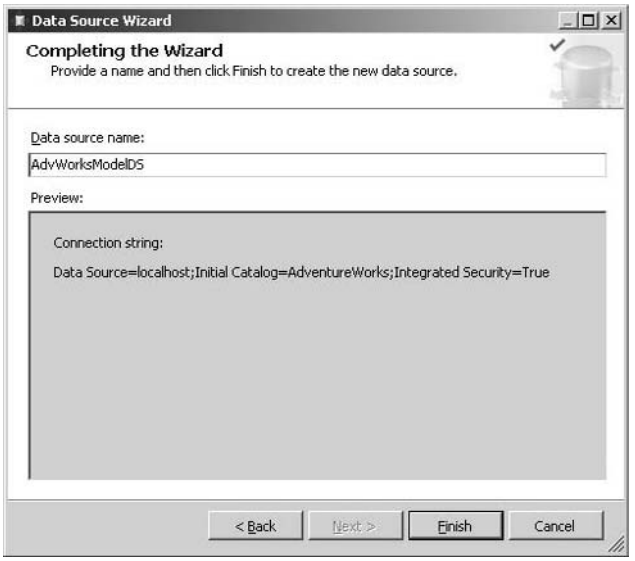
If you went through the steps in Chapter 3, you will have a data source object already. Do not use this object.

4. In the Connection Manager dialog box, type your server name in the Server Name text box.
5. Select Use Windows Authentication.
6. Select AdventureWorks from the ''Select or enter a database name'' in the list box.

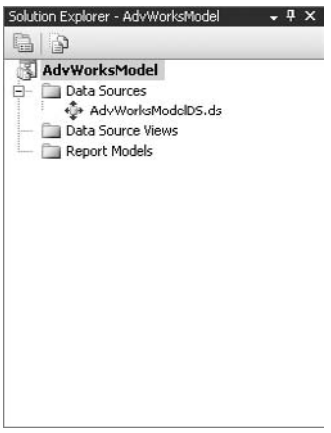


7. Click OK to create the connection.
8. You should now be back at the ''Select how to define the connection'' screen.

9. Verify that the ''Create a data source based on an existing or new connection'' option is selected, verify that *servername*.AdventureWorks is selected in the Data Connections list box and click Next.
10. In the Data source name box, type **AdvWorksModelDS**, and then click Finish.

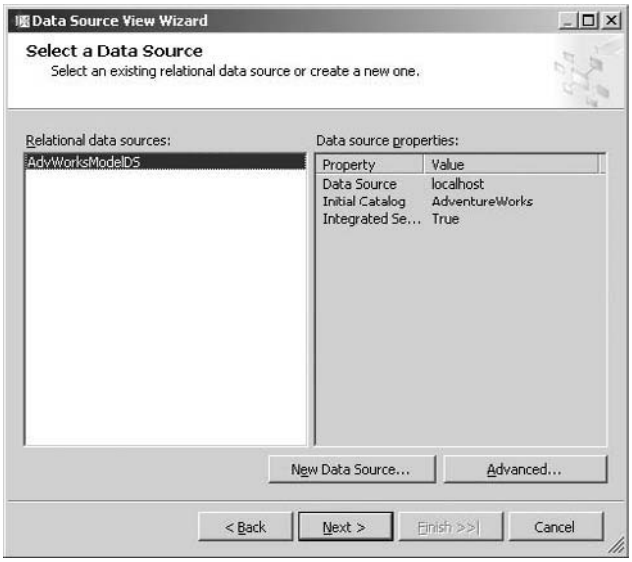


11. You should now see the `AdvWorksModelDS.ds` file in Solution Explorer.

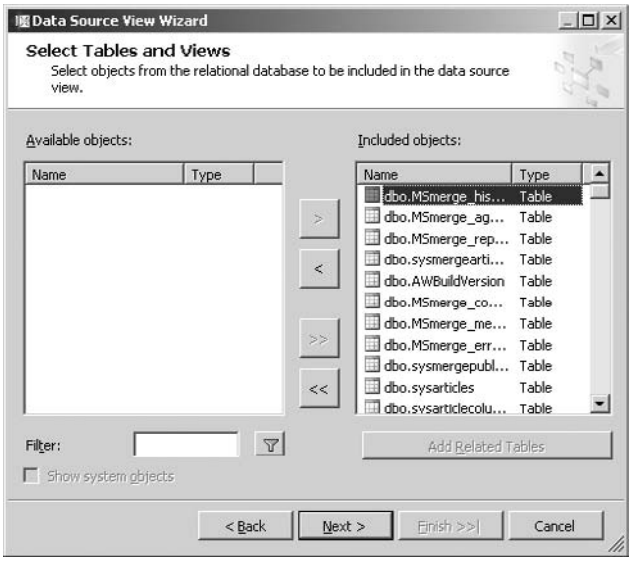


The next step is to create a data source view. Here's how:

1. In Solution Explorer, right-click Data Source Views and click Add New Data Source View.
2. On the Welcome screen, click Next.
3. On the Select A Data Source screen, select the AdvWorksModelDS data source you created and click Next.

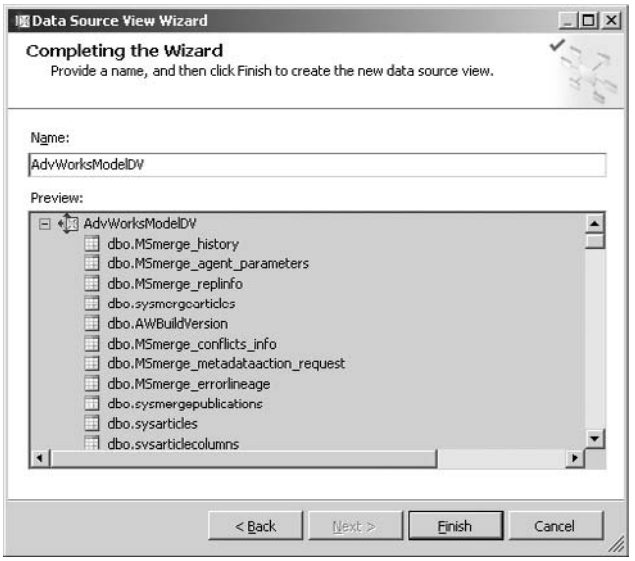


4. On the Select Tables And Views screen, click the≫button to add all available objects to the view.



5. Click Next.

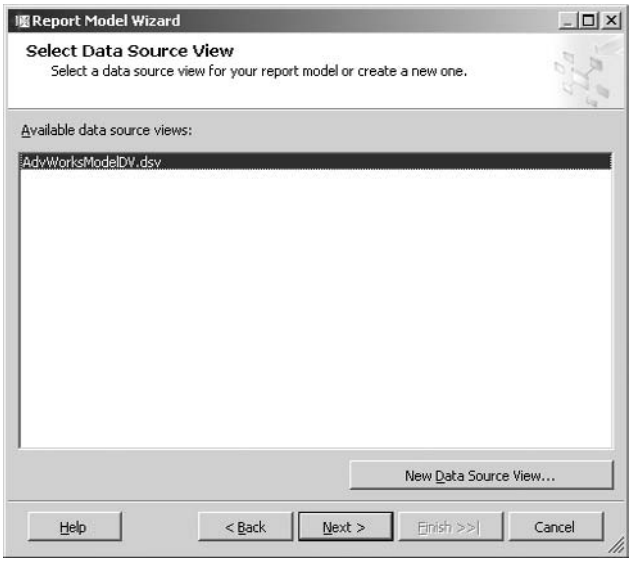
6. Type **AdvWorksModelDV** in the Name text box.



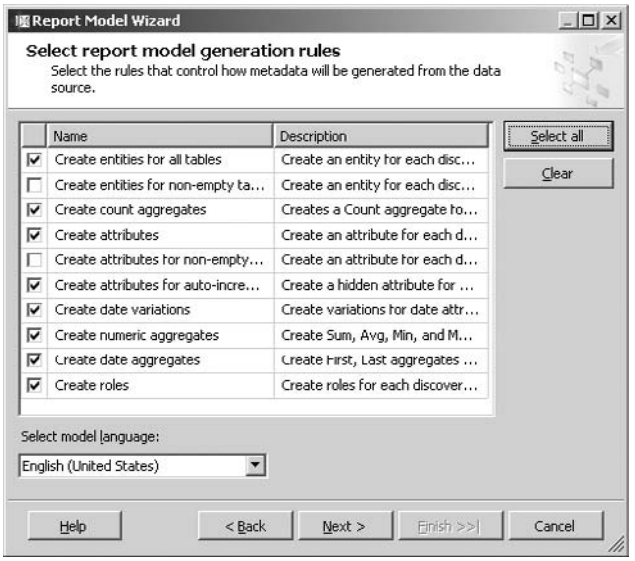
7. Click Finish.
8. You should now see the `AdvWorksModelDV.dsv` file in the Solution Explorer.

Now that you have a data source and a data source view in your project, you are ready to create a report model:

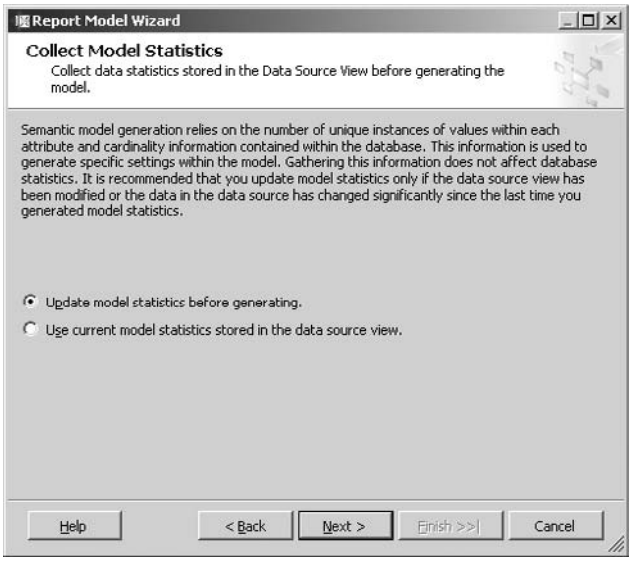
1. In Solution Explorer, right-click Report Models, and then select Add New Report Model.
2. On the Welcome screen, click Next.
3. On the Select A Data Source View screen, select the `AdvWorksModelDV.dsv` file that you just created and click Next.



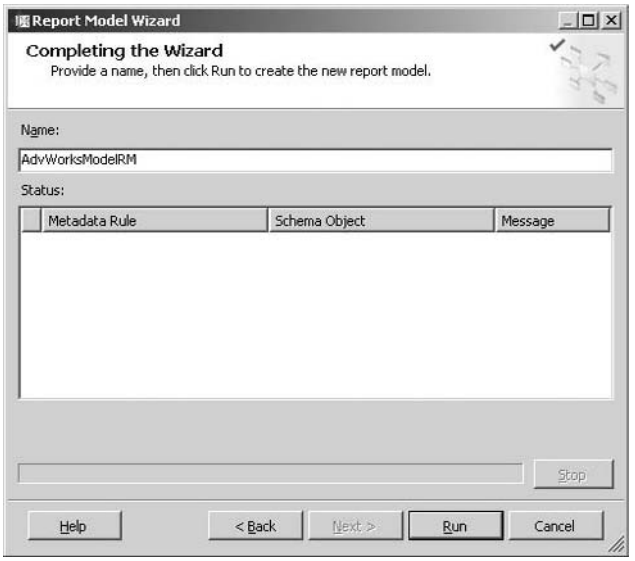
4. The Select Report Model Generation Rules screen is displayed. These rules determine how report model metadata is generated from the data source and combined into objects called entities (more on entities later in this chapter). For now, accept the defaults and click Next.



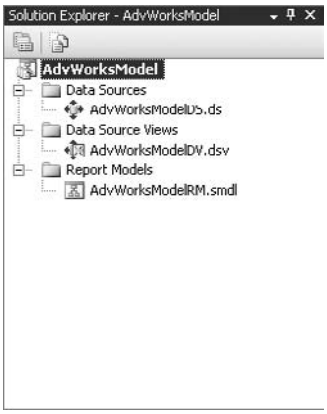
5. On the Update Statistics screen, make sure that Update Model Statistics Before Generating is selected, and then click Next.



6. In the Name box, enter **AdvWorksModelRM** and click Run.

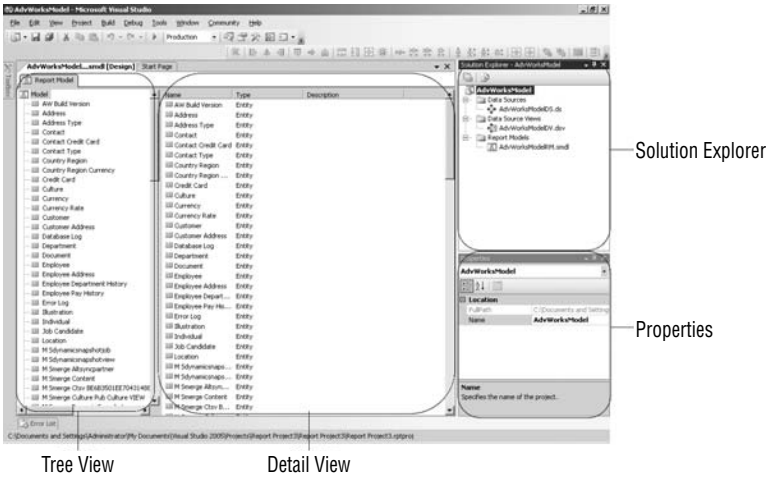


7. To exit the wizard, click Finish.
8. If you are prompted to reload the `AdvWorksModelDV.dsv` file, click Yes.
9. You should now see the AdvWorksModelRM object in the Solution Explorer.



After the wizard is finished, the report model is opened in the development environment, as shown in Figure 8.2. At the far left is the Tree View, which shows a list of all of the folders and entities that make up the model. When you select an object in the Tree View pane, the details are displayed in the Detail View, which is in the middle of the screen. To the far right is the Solution Explorer and Properties window.

FIGURE 8.2
The design environment is composed of the Tree View, Detail View, Solution Explorer, and Properties window.



UNDERSTANDING ENTITIES

I've already mentioned entities twice, and now is the time to explain them. An entity is a logical collection of model items that can be expressed in user-friendly terms. Entities can contain fields, roles, folders, and expressions.

Fields A field is an object that is bound to the source column in the database.

Roles A role defines the relationship between entities that are related to one another. Entities can be related in a one-to-one relationship or a one-to-many relationship. For example, an Employee entity would have a one-to-one relationship with an Address entity because an employee lives at only one address. A Product entity may have a one-to-many relationship with a Store entity because the product is sold at more than one store.

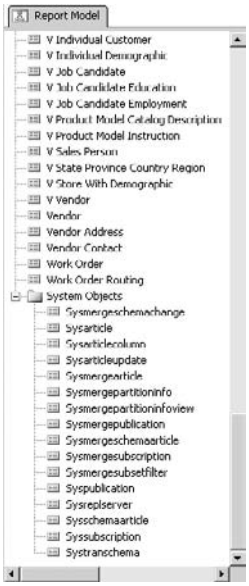
Folders Folders are used for organizing objects in the model. There is more on this later in this section.

Expressions An expression is a field that does not have a value of its own. The value presented from an expression field is actually calculated from the values of other fields. For instance, if you have entities for the total sales of each region, you can create an expression that shows total sales for your entire company by summing the values from each of the regional entities.

Using Folders

Some models can get very complex, with a large number of objects in the Tree View pane, which can become unwieldy. To help with this, you can create folders to organize the project. You can also assign permissions to folders to prevent users from seeing parts of the model. Let's create a folder in the project and move some objects into it:

1. In the Tree View, right-click Model, hover over New, and click Folder.
2. Scroll down to the bottom of the Tree View and find the folder named NewFolder.
3. Right-click NewFolder and click Rename.
4. Type **System Objects** for the name and press Enter.
5. Click the Model node at the very top of the Tree View.
6. In the Detail pane, scroll down until you find the objects that start with sys_.
7. Click Sysarticle.
8. Hold down the Shift key and click Systranschema to select all of the objects that start with sys_.
9. Drag the objects to the System Objects folder in the Tree View.
10. You should now see all of the objects that start with sys_ in the new folder.



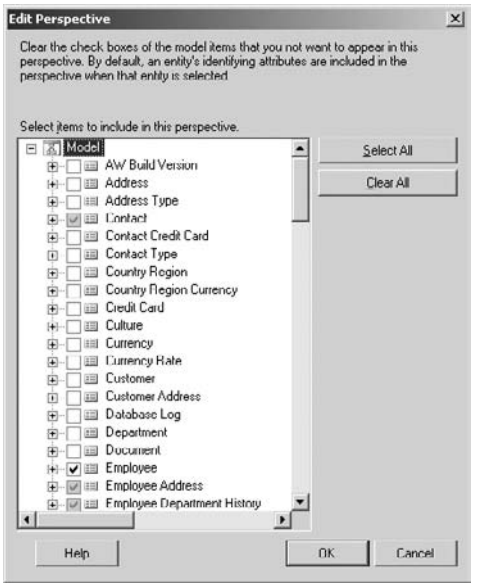
Working with Perspectives

Some report models can get quite large, which makes them difficult for end users to work with. You can make these large models easier to work with by creating perspectives.

A perspective is a subset of a larger model that exists as an object inside the model it is created for. Like models, perspectives can contain roles, entities, folders, expressions, and so on. Unlike models, perspectives are not created using the model wizard; you must create them manually.

Let's create a perspective that contains all the information related to the Employee entity:

1. In the Tree View, right-click Model, hover over New, and click Perspective.
2. In the Edit Perspective dialog box, click the Clear All button; clearing everything will take some time.
3. Check the box next to Employee. Notice that everything related to the Employee entity is automatically checked for you.



4. Click OK to create the perspective.
5. Click the Model node in the Tree View and scroll down in the Detail pane until you find NewPerspective.
6. Right-click NewPerspective and click Rename.
7. Type **EmployeePerspective** for the name and press Enter.

Deploying a Model

Once you have created the model, you must deploy it so that your users can access it. The first time you publish a model, the data source and semantic models are published to the Report Server. On subsequent updates, only the semantic models are deployed.

Before you can deploy your model, you may need to make a change to your server configuration. By default, `ASP.NET` only allows users to upload 4MB of data at once, which

means that you will get an error message if you try to deploy a model larger than 4MB, like AdvWorksModel. To prevent this, you need to change a setting in the `machine.config` file:

1. Open *X*`:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG\machine.config`.
2. Scroll down to the <`system.web`> section and add <**httpRuntime maxRequestLength = "16384"/**>. It should look like this:

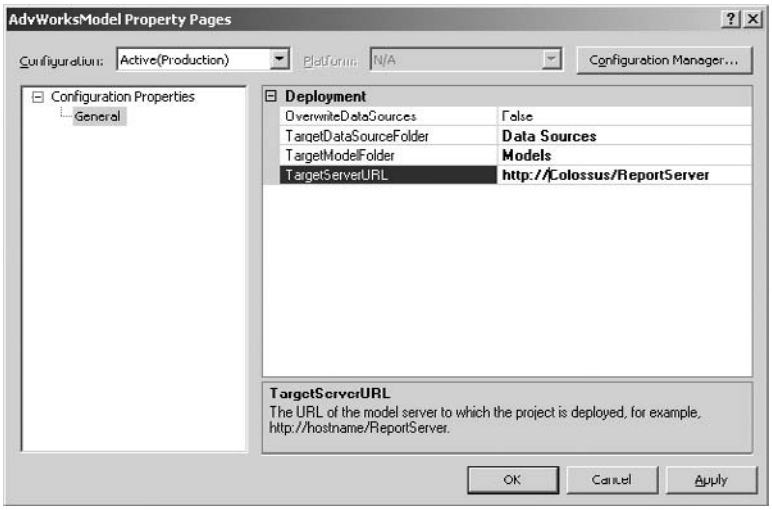
```
<configuration>
...
<system.web>
...
<httpRuntime maxRequestLength=''16384''/>
...
</system.web>
...
</configuration>
```

3. Save and close the file.

Now you can deploy the AdvWorksModel model:

1. In Solution Explorer, right-click AdvWorksModel and click Properties.
2. Change the `TargetServerUrl` property to match your server name: `http://`*`servername`*`/Reports`. If you installed Reporting Services on a named instance the URL would be:

```
http://servername/Reports$InstanceName
```



3. Click OK.
4. From the Build menu, select Deploy AdvWorksModel.
5. A message will be displayed in the status bar when the operation is complete.

Using Report Manager and Management Studio

Under most circumstances, Model Designer is the tool of choice for building models because it offers the most functionality and flexibility. The only problem is that Model Designer does not provide a way to create models based on Analysis Services cubes — which is why we have more than one tool for creating models.

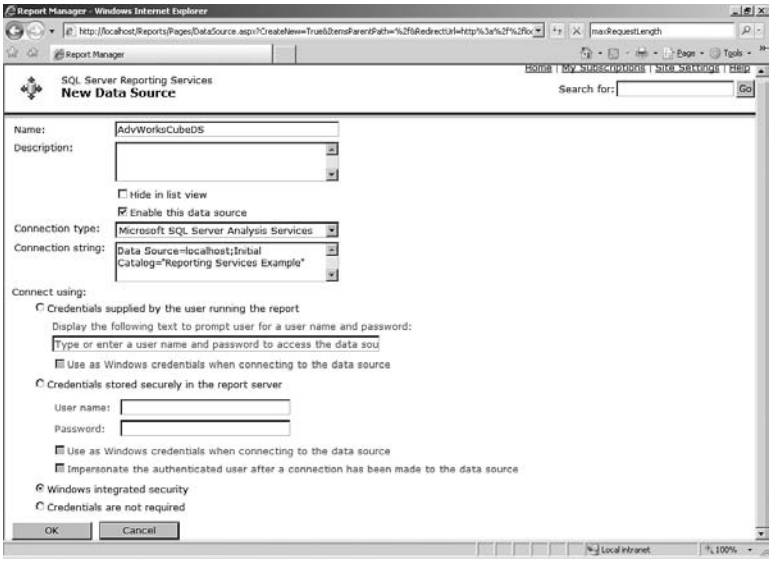
If you need to create a model based on a cube, use either Report Manager or Management Studio. Of course, you can also use these tools to create models based on SQL Server 2000, SQL Server 2005, and Oracle 9.2.0.3 databases as well.

Let's use Report Manager to create a model based on the Reporting Services Example cube created in Chapter 6:

NOTE

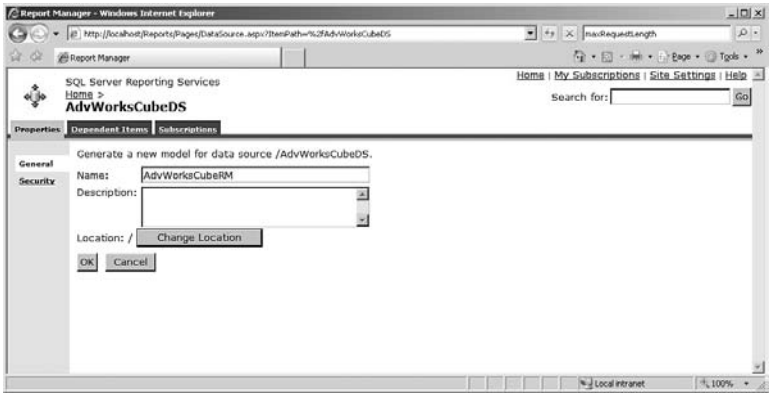
You will not need this model to work with Report Builder later in this chapter, so these steps are optional.

1. Open your Web browser and open `http://localhost/Reports`.
2. Click New Data Source.
3. In the Name box, type **AdvWorksCubeDS**.
4. Make certain that the Enable This Data Source check box is selected.
5. In the Connection Type list, select Microsoft SQL Server Analysis Services.
6. In the Connection String box, enter **Data Source = localhost;Initial Catalog = ''Reporting Services Example''**.
7. Check the Windows Integrated Security option.



8. Click OK to create the data source.
9. In Report Manager, click the link for the new AdvWorksCubeDS data source.
10. Click Generate Model.

11. In the Name box, enter **AdvWorksCubeRM**.



12. Click OK.

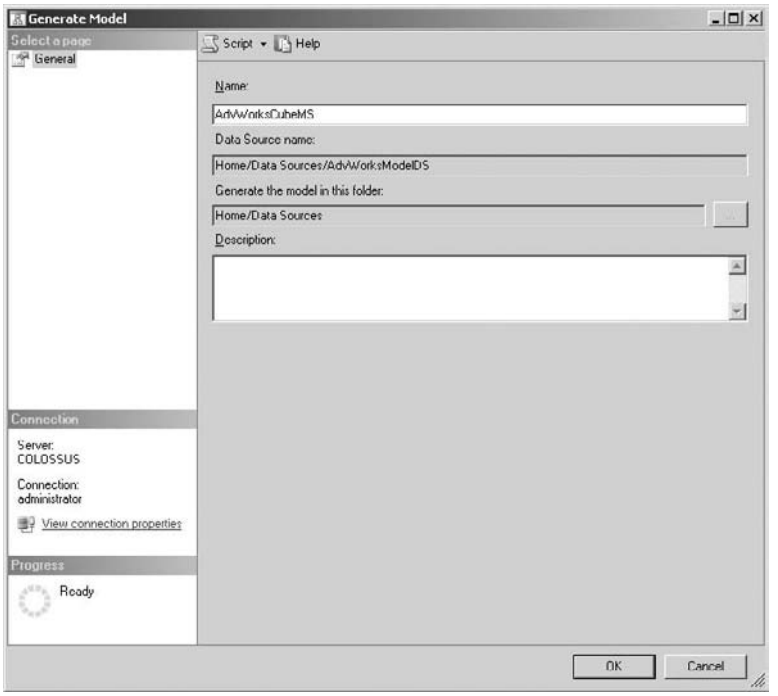
The process for creating a model in Management Studio is similar:

1. Open Management Studio and connect to your Report Server.

2. In Object Explorer, expand Home, then Data Sources.

3. Right-click the AdvWorksCubeDS data source and select Generate Model.

4. On the Generate Model page, in the Name box enter **AdvWorksCubeMS**.



5. Click OK.

Using Report Builder

Once you have created and published some report models, your users can start creating ad hoc reports using Report Builder. Using this powerful tool, end users can design and create reports without needing to know the intricacies of the underlying data sources.

Unlike Report Manager, Report Builder is a Windows application that runs on the client's machine, not in a web browser. So, before they can do anything with this tool, your users will have to install it on their machines. The easiest way for users to install Report Builder is by opening it from Report Manager, like this:

1. Open a web browser and go to the URL for your Report Server, which is usually `http://`*`servername`*`/`reports.
2. Click Report Builder to open the Report Builder application.

You can also open Report Builder by going directly to the application's URL. There are two direct URLs: one for partial trust and one for full trust. Just open the correct one in your web browser.

- Full trust: `http://`*`servername`*`/reportserver/reportbuilder/reportbuilder.application`
- Partial trust: `http://`*`servername`*`/reportserver/reportbuilder/reportbuilderlocalintranet.application`

Follow these steps to start Report Builder in SharePoint integrated mode:

1. Navigate to the site that contains the library you want and open the library.
2. On the New menu, click Report Builder Report to open Report Builder.

Creating and Designing Reports

With Report Builder installed and opened, your users can start building reports. First, the user needs to create a report, which involves selecting a data source and report layout that the new report will be based on.

A report layout helps the user get started by providing a basic format, or template, for the report. Out of the box, there are three report layout options for you to choose from:

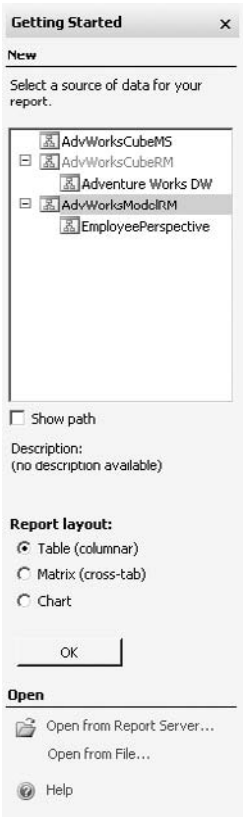
Table The table layout is the most basic report layout. This layout shows a table in which data from the data source is presented in rows and columns, much like an Excel spreadsheet.

Matrix Like the Table layout, the Matrix layout is arranged in columns and rows. However, with the Matrix layout you can present static or dynamic columns and rows. When the user runs the report, the dynamic columns will expand across the page, presenting as many columns as there are groups in the report. Dynamic rows expand down the page. The data presented to the user are summary values of the data based on the intersection of rows and columns. In this way, a matrix layout works much like a pivot or cross-tab table in Excel. This is useful for creating a report based on an Analysis Services cube.

Chart As the name implies, the chart layout presents a graphical chart to the user viewing the report. The individual creating the report can choose from several different styles, such as area, bar, column, doughnut, line, and pie. Charts also allow a variety of formatting options, like color and 3D effects.

Let's create a new report now:

1. Select File ➢ New.
2. In the Getting Started pane, under Select A Source Of Data For Your Report, click the AdvWorksModelRM model.
3. Select Table (columnar)under Report layout.

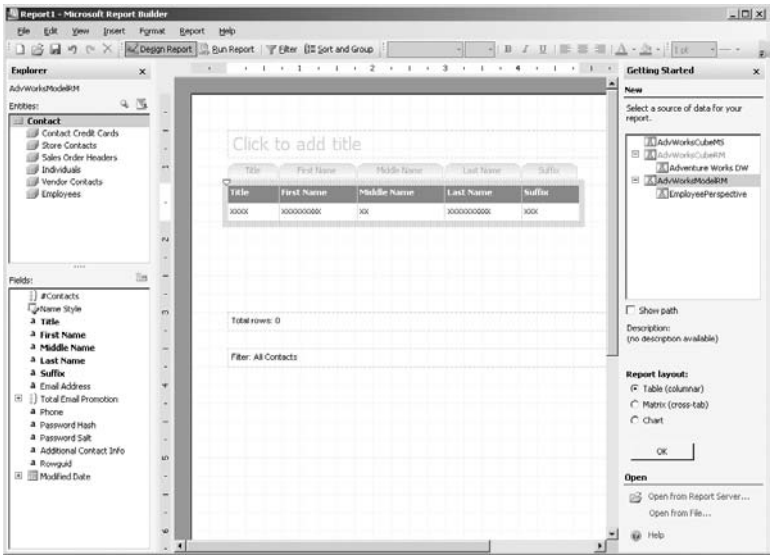


4. Click OK to close the Getting Started pane and open the report layout in the design area.

The new report you have just created is a blank slate; to make it useful you need to add objects to it, such as fields, labels, images, and the like. To get a feel for how this works, you will drag some objects onto the design surface now:

1. Click Contact in the Entities list on the right.
2. Drag the Title field from the Fields list to the area of the design surface labeled ''Drag and drop column fields.''
3. Drag the First Name field to the right of the Title field on the design surface; do not drop the field until you see a blue I-beam appear on the surface. The I-beam tells you that it is safe to drop the field in that location on the surface.

4. Drag the Middle Name field to the right of the First Name field on the design surface, and drop it when you see the blue I-beam.
5. Drag the Last Name field to the right of the Middle Name field on the design surface, and drop it when you see the blue I-beam.
6. Drag the Suffix field to the right of the Last Name field on the design surface, and drop it when you see the blue I-beam.



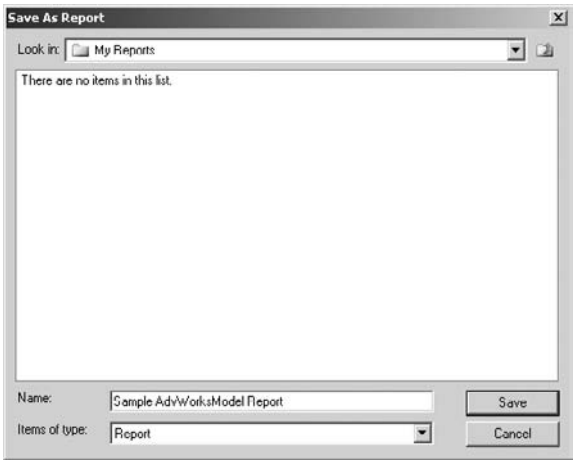
7. To run the report, click Run Report on the toolbar.

The reason you need to run the report is because in design mode, you do not see the actual data in the report. You see Xs in place of character data and 0s in place of numerical data. Running the report fills the report with live data for you to work with. When you click the Run Report button, you switch from design mode to run mode.

While in run mode you can continue to work with the report. You cannot move the fields around on the page as you can in design mode, but you can filter, sort, and group the data to control how it is displayed on the page. You can print and export the report if you are happy with it.

When you have finished working with the report, you can save it to the Report Server. Let's do that now:

1. Select File ➢ Save.
2. In the Name box, type **Sample AdvWorksModel Report**.
3. In the list of folders, double-click My Reports.



4. Click Save.
5. Open Report Manager from the `http://localhost/Reports` URL.
6. Click the My Reports link. You should see the Sample AdvWorksModel Report listed.

Real World Scenario

Case Study: Chocolaty Goodness Candies

When I worked for them, CGC was a mid-sized company with a sales staff of about 50 people. Every week, the sales staff met with clients at their location, and they needed a custom report developed so they could take the latest sales figures with them in the format of the week.

When we were using SQL Server 2000 Reporting Services, this added quite a bit to the development workload because we had to create each report manually and make sure that it met the salesperson's needs.

When we migrated our report system to SQL Server 2005 Reporting Services, we quickly saw the advantages of allowing the sales staff to use ad hoc reporting. We gave the sales staff the permissions they needed to access Report Builder, created a report model based on the Sales database, and let the sales staff loose. It only took them a few weeks to get used to the new application, and when they settled in the custom development requests dropped by about 80 percent.

The Bottom Line

Configure your systems to support ad hoc reporting. Before your users can start creating ad hoc reports, you need to configure the systems to run Report Builder.

Master It You have installed Reporting Services on your network and you need to allow the sales staff to create ad hoc reports using Report Builder. All of the sales staff employees are members of a Windows security group named Sales in the Corp domain. What do you need to do to allow the sales staff access to Report Builder?

Create and deploy report models. A report model is a layer of metadata in between the database and Report Builder that is used to describe the underlying data source in user-friendly terms. Models can be based on SQL Server 2000 databases, SQL Server 2005 databases, Oracle 9.2.0.3 and greater databases, and SQL Server 2005 Analysis Services cubes. You can use Model Designer, Report Manager, and Management Studio to create report models; the only caveat is that Model Designer cannot be used to create models based on Analysis Services cubes.

Master It Now that your sales staff has access to Report Builder, they want to be able to create reports based on the data in the Sales database. How would you create a model that presents all of the data in the Sales database for them to use?

Use Report Builder to work with ad hoc reports. Report Builder is a `.NET` Framework 2.0 application that users can install using ClickOnce technology and run on their local machine to create and design reports. The reports they design can be saved to the server for future use, or discarded after a single session.

Master It To create a report you need to choose a report layout. What default report layouts are available?

Chapter 9

Subscriptions

As your users become more familiar with Reporting Services and start to realize its true potential to help them perform their duties, they will rely on it more and more. They will use reports to make serious decisions regarding how to run the business, what products to sell, how many new employees to hire, and so on. This raises a new issue: report availability.

Users are not always at the office when they need to make a decision. For example, a salesperson who travels to client sites regularly needs reports to help make sales. Also, executive staff may travel frequently to expand the business and they will need reports to help them make decisions while they are on the road. There are many circumstances like these where users will need reports when they do not have access to the Report Server. To get the data they need, these mobile users can create subscriptions.

A *subscription* is a request to deliver a report to a user or group of users at a specific time or in response to an event on the server. The subscription can deliver the report in a number of ways and in a variety of formats.

Two types of subscriptions are available for your use: standard and data driven. A standard subscription is composed of static values that do not change during execution. Individual users create standard subscriptions for themselves. With data-driven subscriptions, the presentation, delivery, and parameter values are all stored in a data source so they can all be changed at runtime. Typically, Report Server administrators or developers create data-driven reports.

The reports created by subscriptions can be saved to a file on disk and the users can access them over the company network, or they can be e-mailed to the users. You can also use the special Null Delivery Provider extension to preload the report cache.

Report Server administrators also need to manage and monitor the subscriptions on their servers. They must check subscriptions to see if they are running at the right times, and to see whether they are still running at all. If the subscription is inactive, the administrator may have to delete it from the server.

In this chapter you will learn:

- How subscriptions are processed
- How to create standard subscriptions
- How to create data-driven subscriptions
- How to manage subscriptions
- How to use subscriptions to preload the report cache

Understanding Subscriptions

If you subscribe to a newspaper, you expect to get a paper every day by a certain time because you have a standing request for the paper to be delivered. A Reporting Services subscription is similar; it is a standing request for a report to be delivered based on a timed schedule or an event on the server. A subscription is composed of several parts:

- A report that has stored credentials or that does not require credentials, so that it can run unattended
- A delivery method with corresponding settings
- A rendering extension to render the report in the requested format
- Conditions for processing the report, such as the time of day
- Any parameters that may be required to run the report

You do not need to configure the server to allow users to create subscriptions; that functionality is configured by default. The security settings are not configured by default, so you may need to make some changes to the security settings. Two permissions are specific to subscriptions:

- The Manage Individual Subscriptions task allows users to create and modify a subscription for an individual report. As the name implies, users can modify only subscriptions that they have created.
- The Manage All Subscriptions task allows users to access and modify all subscriptions. This is required for users to create data-driven reports.

If you do not want some users to create subscriptions, remove the Manage Individual Subscriptions task from the role they are assigned to. Bear in mind, though, that if you remove this task after the user has already created subscriptions, the existing subscriptions will continue to execute until they are deleted but the user will not be able to create new subscriptions.

NOTE

See Chapter 3 for a discussion of tasks and role assignments.

Two delivery extensions are available out of the box: File Share and Email Delivery. To disable one of these extensions, you need to modify the `RSReportServer.config` file. If you decide to remove an extension from the server, be sure to delete any subscriptions that may rely on it or they will cause errors on the server every time they are executed.

Reporting Services contains a scheduling and delivery processor component to make sure the reports are delivered on schedule. Here is how the process works:

1. When a user creates a subscription, they define a schedule, which defines the date and time that the report will be run, a snapshot will be refreshed, or a cache will expire.
2. The Report Server saves the schedule information in the Report Server database.
3. The Report Server creates a job for the SQL Server Agent to run, based on the schedule information provided.
4. When the SQL Server Agent runs the job on the date and time specified in the schedule, it creates an event that is added to a queue maintained by Reporting Services.
5. The Report Server Windows service polls the queue for new events every 10 seconds and processes new events as they are entered.

From this description, you can see that the scheduling and delivery component requires that the Report Server Windows service and the SQL Server Agent both be running. If the SQL Server Agent is stopped, no new events will be added to the queue and subscriptions will not be processed. If the Report Server Windows service is stopped, the SQL Server Agent will continue to add new events to the queue but they will not be processed right away. As soon as the Report Server Windows service is restarted, it will process all the events that have piled up in the queue.

Configuring the Servers

You need to configure the servers before your users can start creating subscriptions. For file-share delivery, you need to make sure that a file share is available on the network. For e-mail delivery, you must configure the servers to work with the e-mail system.

Configuring a File Share

If your users are in the office most of the time or have access to the network over a virtual private network (VPN) when they are not in the office, you may want to have them create file-share subscriptions. A file-share subscription generates a report that is saved in a folder that is shared over the network. This can take up some disk space, but unlike the e-mail delivery, it does not consume space in the user's mailbox. You can also delete files from the file share when they are no longer needed to save disk space.

To allow users to create subscriptions with file-share delivery, create a file share on the network and make sure that the appropriate permissions are assigned. When a subscription created with the file-share delivery option is run, it connects to the network share using the credentials specified in the report. So you need to make sure that the account used by the subscription has write permissions on the folder that is being shared. Also, the users who will read the report need read access to the folder that is being shared.

Follow these steps to create a network share to be used for the remainder of this chapter:

1. In Windows Explorer, create a new folder named **Subscriptions**.
2. Right-click the folder and select Properties.
3. On the Sharing tab, click Share This Folder.
4. Accept the default share name, which is the same as the folder name.
5. Click Permissions.
6. Click Add.
7. Subscriptions that deliver reports to a file share connect to the share with the username and password of an account that you specify when creating the subscription. That account must have write access to this folder. For now, enter your own account and click OK.
8. Select your account and grant Full Control permission.
9. Make sure that the users who will read the reports in this folder have read access. For now, accept the default of Everyone and click OK.

Configuring E-mail Delivery

If your users do not have access to the network when they are out of the office, or if they prefer to access reports from their inbox for organizational purposes, you can let them create

e-mail subscriptions. An e-mail subscription generates a report that is delivered directly to the user's e-mail inbox. This takes up space on the mail server, but it does not take up space on the network file servers and it is easier for many users to find because they check their e-mail frequently throughout the day.

The e-mail delivery extension is not configured by default, so you must make some configuration changes to get it running. For basic functionality, you will need to tell the Report Server which e-mail server to use. There are some more advanced options available for you to configure if necessary.

One of the settings you may need to modify is the To field. When a subscription e-mails a report to a user, the value in the To field is created by using the domain user account of the recipient. For example, if your network username is MyCorp\Jjohnson, the e-mail recipient would be Jjohnson. If you are using a different naming scheme for your e-mail accounts, this default setting will not work for you. You can change this by opening the `RSReportServer.config` file and setting `SendEmailToUserAlias` to false and `DefaultHostName` to the DNS name or IP address of the SMTP server or forwarder. Doing so will allow the user creating the subscription to enter a custom value in the To field.

You can also configure Reporting Services to use either a local or remote SMTP server. It is usually best to use a remote server to offload the mail-processing overhead. You may have to modify some settings if you have nonstandard configuration settings on your SMTP server.

For instance, SMTP uses port 25 by default; if your server is configured to use another port, you must change the `SMTPServerPort` setting in the `RSReportServer.config` file.

Also, the Report Server connects to the SMTP server using anonymous access, which is fine for sending most e-mails. If you need to send reports to restricted distribution lists, you will have to change `SMTPAuthenticate` from 0 to 2 in the `RSReportServer.config` file.

You may also need to set the name that shows up in the From field of the e-mail that is sent to the user. You can configure this setting using the `From` value in the `RSReportServer.config` file.

To configure the server for remote SMTP access using the Reporting Services Configuration tool:

1. Make certain that the account that the Report Server Windows service uses has Send As permissions on your SMTP server.
2. Start the Reporting Services Configuration tool and connect to the Report Server instance.
3. On the Email Settings page, enter the name of your SMTP server.
4. In Sender Address, enter the name of an account that has permission to send e-mail from your SMTP server. For this exercise, you may want to enter your own account information, though that is not recommended for a production system.
5. Click Apply.

To configure the more advanced settings for e-mail delivery:

1. Open the `RSReportServer.config` file in a text editor.
2. In the Delivery section, find the `ReportServerEmail` section.
3. Verify that the `SendUsing` setting is set to 2, which means that the Report Server is using a remote SMTP server.
4. Change the `SMTPAuthenticate` setting to 2 to allow sending e-mail to restricted distribution lists.

5. Change the `From` setting to an account that has permission to send e-mail from the SMTP server.
6. Save the file. The Report Server will use the new settings automatically; you do not need to restart the service.

If your Report Server is in another domain or behind a firewall, you may need to use a local SMTP server. Here is how to configure the Report Server to use a local SMTP server (do not do this now unless you do not have access to a remote SMTP server):

1. First, you need to install an SMTP server. On Windows Server 2003, open Add Or Remove Programs in Control Panel.
2. Click Add/Remove Windows Components to start the Windows Component Wizard.
3. Select Application Server and click Details.
4. Select Internet Information Services (IIS) and click Details.
5. Select the SMTP Service check box and click OK.
6. On the Windows Component Wizard, click Next, then click Finish.
7. Verify that the service is running in the Services console.
8. Open the `RSReportServer.config` file in a text editor.
9. If it is not already configured, make sure that the `UrlRoot` setting is set to the Report Server URL address.
10. In the Delivery section, find the `ReportServerEmail` section.
11. Clear any values in the `SMTPServer` setting, but do not delete the tags.
12. Set the `SendUsing` setting to 1, which means that it is configured to use a local SMTP server.
13. Set `SMTPServerPickupDirectory` to a folder on the local drive, usually `C:\Inetpub\mailroot\Drop`.
14. Set the `From` setting to an account that has permission to send e-mail from the SMTP server.
15. Save the file.

Configuring the Default Delivery Extension

When your users create subscriptions using Report Manager, they are presented with a drop-down list of all the available delivery extensions to choose from. If your server is not configured for e-mail delivery, the file-share extension is the default choice. If your server is configured for e-mail delivery, e-mail delivery becomes the default.

If you configure file delivery first and configure e-mail delivery later, e-mail will take over as the default delivery extension. You may not want this to happen if your users are already accustomed to the file-share extension as the default, so you can change the default setting. Here's how:

1. Open the `RSWebApplication.config` file in a text editor.

2. Change the Report Server `FileShare` section as follows:

```
<ExtensionName="ReportServerFileShare" Type="Microsoft.ReportingServices.➥
FileShareDeliveryProvider.FileShareUIControl,
   ReportingServicesFileShareDeliveryProvider">
<DefaultDeliveryExtension>
True
</DefaultDeliveryExtension>
</Extension>
```

3. Change the `DefaultDeliveryExtension` setting of the Report Server `Email` section to False.
4. Save the file.

Creating Subscriptions

Users can create one of two subscription types: standard and data driven. You can also integrate subscriptions with SharePoint if you have SharePoint and SQL Server 2005 Service Pack 2 installed.

The process for creating each type of subscription differs, so in this section we will discuss how to create standard and data-driven subscriptions and then show you how to integrate them with SharePoint.

For the remainder of this chapter we are going to work with the sample reports that come with SQL Server 2005. We'll assume that you have them installed already, just not deployed. Follow these steps to deploy the sample reports on SQL Server 2005 Express Edition with Advanced Services:

1. Open Business Intelligence Development Studio and choose File ➢ Project/Solution.
2. Browse to `C:\Program Files\Microsoft SQL Server\90\Samples\Reporting Services\Report Samples\AdventureWorks Sample Reports`, select `AdventureWorks Sample Reports.sln` and click Open.
3. The default settings for the sample application will not work, so you have to make some changes. Start by choosing Project ➢ Properties.
4. Select True in OverwriteDataSources.
5. Enter this URL in the TargetServerURL text box: `http://localhost/reportserver\$SQLExpress`.
6. Click OK to close the Solution Property Pages dialog box.
7. In Solution Explorer, under Shared Data Sources, double-click the `AdventureWorks.rds` file.
8. In the Shared Data Source dialog box, on the General tab, modify the connection string to use the following value: **Data Source = localhost\SQLExpress; Initial Catalog = AdventureWorks**.
9. Click OK to close the Shared Data Source dialog box.
10. On the standard toolbar, select the Production solution configuration.
11. Click Deploy AdventureWorks Sample Reports on the Build menu to deploy the reports to the local server.

Follow these steps to deploy the sample reports on other editions:

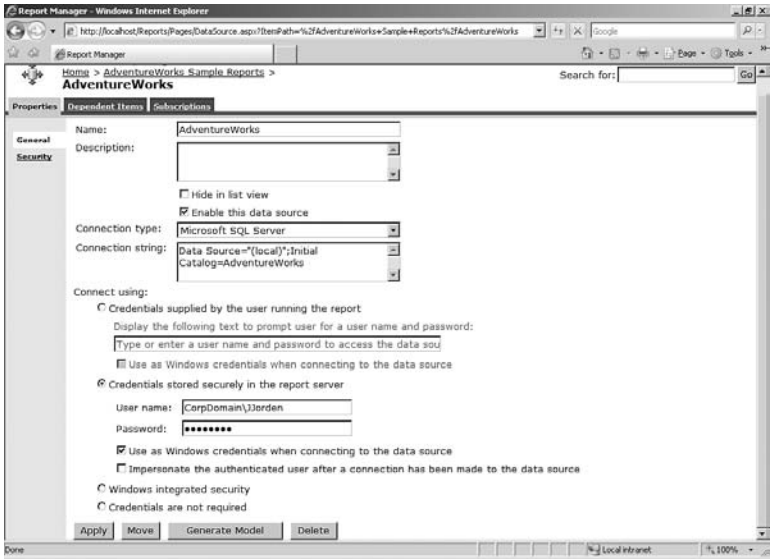
1. Open Business Intelligence Development Studio and choose File ➢ Project/Solution.
2. Browse to `C:\Program Files\Microsoft SQL Server\90\Samples\Reporting Services\Report Samples\AdventureWorks Sample Reports`, select `AdventureWorks Sample Reports.sln`, and click Open.
3. On the standard toolbar, select the Production solution configuration.
4. Click Deploy AdventureWorks Sample Reports on the Build menu to deploy the reports to the local server.

Standard

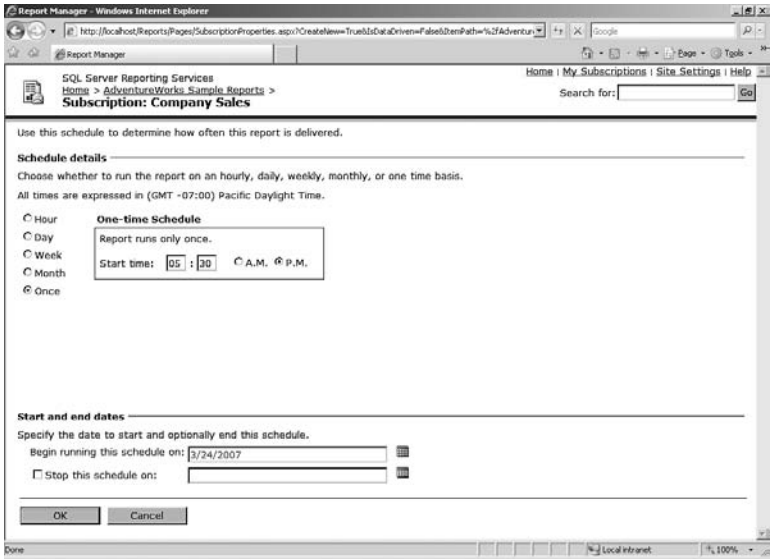
Every report that a user views has some options associated with it, such as the format of the report and any parameters values that may be needed. These options are filled in by the subscription when it creates the report. With a standard subscription, these values do not change at runtime. The subscription creates a single instance of the report for delivery to the users. This means that all the users who receive a report generated by a standard subscription get the exact same data in the exact same format and layout. When an end user creates a subscription, it is usually a standard subscription.

In these steps, you will create a standard subscription and configure it to use file-share delivery:

1. First, configure the data source for the report to use stored credentials; in Report Manager, on the Home page, click the AdventureWorks Sample Reports link and open the AdventureWorks data source.
2. Select the Credentials Stored Securely in the Report Server option and enter your username in the User Name box in the ***domain\username*** format. Then enter your password in the Password text box.
3. Check the ''Use as Windows Credentials When Connecting to the Data Source'' box.

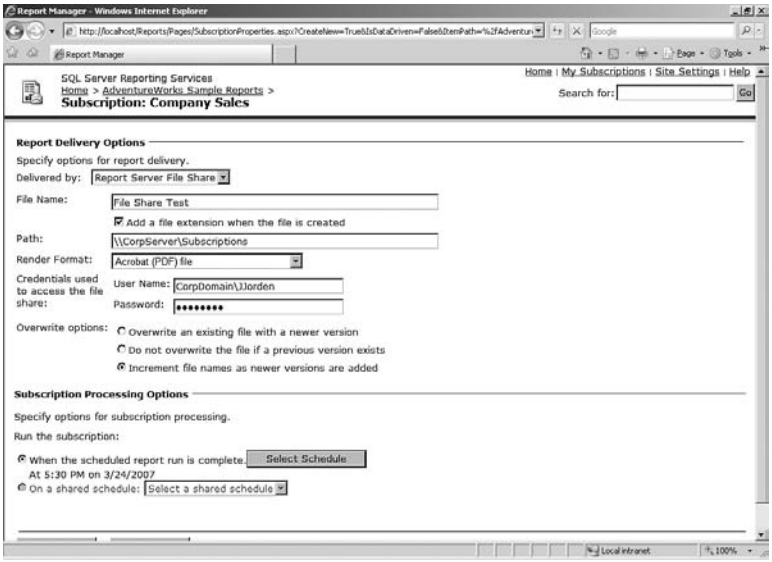


4. Click Apply.
5. Click the Home link at the top right of the page to return to the Report Manager home page.
6. Click the AdventureWorks Sample Reports link and open the Company Sales report.
7. Click the Subscriptions tab, and then click New Subscription.
8. For the method of delivery, select Report Server File Share from the Delivered By list box.
9. Type **File Share Test** in the File Name text box.
10. Ensure that ''Add a File Extension When the File Is Created'' is selected. This causes the report to be saved with a three-character file extension that is determined by the selected rendering format.
11. Select Adobe Acrobat (PDF) from the Render format drop-down list. This will require you to download the free Adobe Reader from `http://www.adobe.com/products/acrobat/readstep2.html`.
12. In the Path text box, type a UNC path to the Subscriptions folder you shared earlier in the chapter. It should look like this *server***Subscriptions**.
13. Select the Increment File Names As Newer Versions Are Added option to leave existing report files intact and create new files when the subscription runs.
14. Click the Select Schedule button to schedule the report.
15. On this screen, you can define how often a report is executed and delivered. For this exercise, select the Once option.
16. In the Start Time, enter a time that is 10 minutes from the current time.



17. Click OK to create the schedule and return to the previous page.

18. In the User Name and Password text boxes, enter your own username and password using the format ***domain\username*** for the username.



19. Click OK to create the subscription.

20. Wait until the subscription is scheduled to run and check the Subscriptions folder for a new file named `File Share Test.pdf`.

Let's also create a subscription using e-mail delivery:

1. On the Home page in Report Manager, click the AdventureWorks Sample Reports link and open the Company Sales report.

2. Click the Subscriptions tab, and then click New Subscription.

3. For the method of delivery, select Report Server E-Mail from the Delivered By list box.

4. If you have configured Reporting Services to allow you to specify a To address, type your e-mail address in the To text box. You can enter multiple e-mail addresses by separating them with a semicolon (;).

5. Check the Include Report box to attach the report.

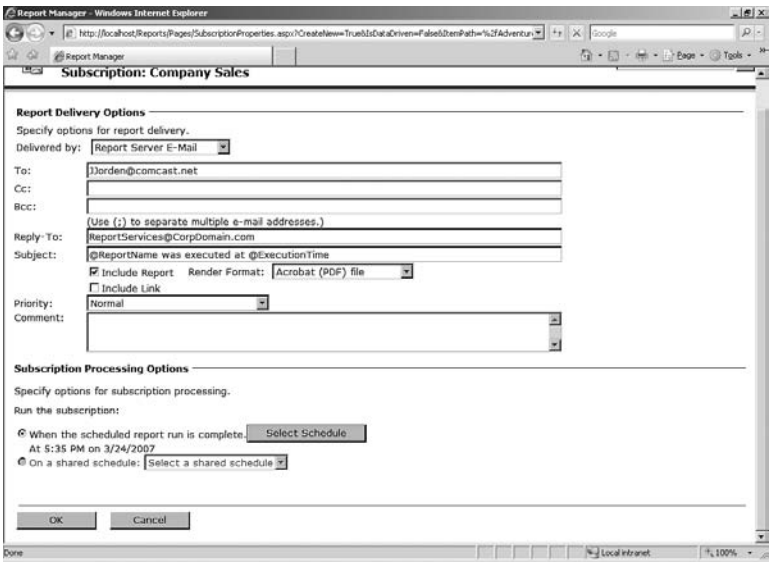
6. Deselect the Include Link checkbox.

7. Select Adobe Acrobat (PDF) from the Render format drop-down list. This will require you to download the free Adobe Acrobat reader from `http://www.adobe.com/products/acrobat/readstep2.html`.

8. Make sure Normal Priority is selected in the Priority list box.

9. Click the Select Schedule button to schedule the report.

10. Select the Once option.
11. In the Start Time, enter a time that is 10 minutes from the current time.
12. Click OK to create the schedule and return to the previous page.



13. Click OK to create the subscription.

Data Driven

Unlike standard subscriptions, data-driven subscriptions are capable of creating multiple reports in multiple formats, each having a unique set of data. For example, you could create a data-driven subscription to deliver a report in PDF format to your Accounting department and in HTML format to your Executive staff. If you are creating a subscription for a report that accepts parameters, you can use different parameter values for each recipient of the report, generating a unique report for each recipient.

This is possible because data-driven subscriptions retrieve subscription settings from a data source, so they can change on the fly. By storing this data in a separate data source and obtaining it at runtime, the data-driven subscription allows you to change settings such as the rendering format, e-mail address, subscriber name, and parameter values each time the subscription is run.

Data-driven subscriptions have a few more components than standard subscriptions. They are made up of:

- The report for which the subscription is defined.
- The delivery extension used to distribute the report.

- The connection string for the subscriber data source. The data source contains the subscriber data that you enter when you define the subscription. This cannot be specified dynamically at runtime.
- The query used to select subscriber data. This cannot be changed at runtime.

You should keep in mind two important restrictions when creating data-driven subscriptions:

- You must use Enterprise Edition because it is the only production edition that supports data-driven subscriptions. Evaluation and Developer editions support data-driven subscriptions as well, but they are not licensed for production use.
- Data-driven subscriptions are not available in SharePoint integrated mode, regardless of the edition used.

When you create the subscriber data source, you need to use a data source that allows schema-only query execution. A schema-only query is a special query that does not return the actual data in the table, just information about the table itself. Reporting Services executes a schema-only query in the early stages of report execution. These are the recommended data sources to use for subscription data:

- SQL Server relational databases
- Oracle databases
- Analysis Services multidimensional and data mining data sources
- XML data sources

You need to increase the query timeout settings in the subscription if you decide to use the XML data processing extension for subscriber data. This is because the XML data processing extension uses milliseconds rather than seconds for query timeout values. The subscription may fail due to insufficient processing time if you don't increase the value.

Also, stored credentials are recommended when using the XML data processing extension to retrieve subscription data at runtime. So avoid using the Credentials Are Not Required option when you configure the subscriber data source connection.

Keep these points in mind when you create the query to retrieve the subscriber data:

- You can only create one query for the subscription.
- The result of the query must contain all the information necessary to run the report, including recipients, parameter values, and rendering formats.
- The subscription will create a report delivery for each row in the result set returned by the query. If there is only one row in the result set, the subscription will create one delivery. If there are 700 rows in the result set, the subscription will create 700 deliveries. So, be sure to verify that your query is returning the expected number of rows before putting it into production.

Now you are ready to try your hand at creating a data-driven subscription. Before you can create the subscription, create a table to hold the subscriber data that the subscription uses to customize the reports. In the following steps, you will create a database named SubscriberData and add a table to hold subscriber data:

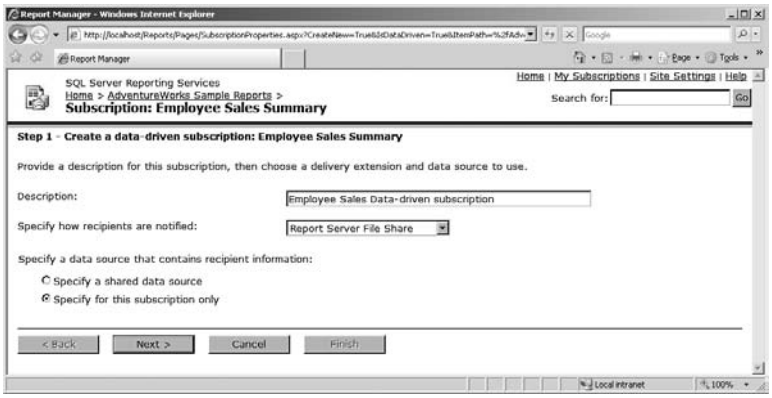
1. Start Management Studio, and open a connection to a Database Engine.
2. Right-click Databases and click New Database.
3. In the New Database dialog box, type **SubscriberData** in the Database Name text box and click OK.
4. Click the New Query button on the toolbar.
5. Enter this query:

```
USE SubscriberData
CREATE TABLE [dbo].[UserInfo]
(
[SubscriptionID] [int] NOT NULL PRIMARY KEY,
[EmployeeID] [int],
[LastName] [nvarchar] (50) NOT NULL,
[FileType] [bit],
[Format] [nvarchar] (20) NOT NULL,
) ON [PRIMARY]
GO
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('1', '289', 'Valdez', '1', 'IMAGE')
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('2', '284', 'Alberts', '1', 'MHTML')
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('3', '275', 'Blythe', '1', 'PDF')
GO
```

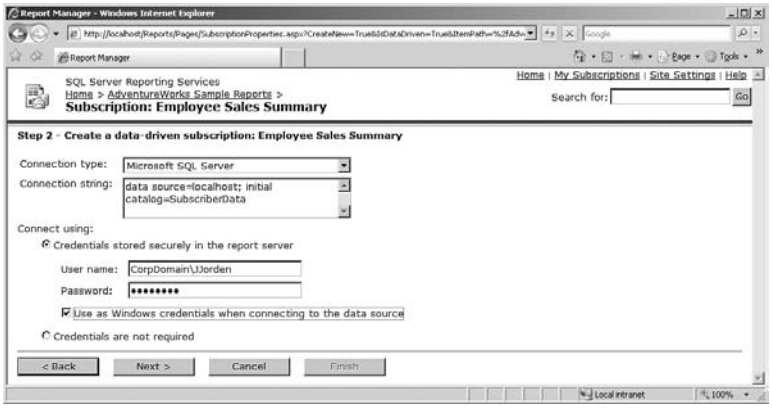
6. Click the Execute button on the toolbar.

Now that you have a database and table to hold the subscriber data, you can create a data-driven subscription based on the Employee Sales Summary report:

1. On the Home page in Report Manager, click the AdventureWorks Sample Reports link and open the Employee Sales report.
2. Click the Subscriptions tab, and then click New Data-Driven Subscription.
3. Type a description for the subscription if you like; this step is optional.
4. Select Report Server File Share from the delivery method drop-down list.
5. Make sure that Specify For This Subscription Only is selected for the recipient information data source and click Next.

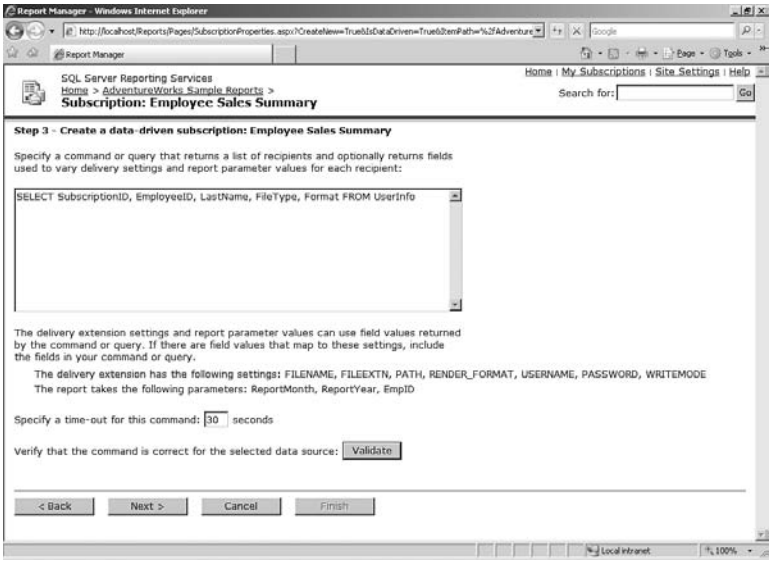


6. In Type, select Microsoft SQL Server as the data source type.
7. In Connection string, type the following connection string: **data source = localhost; initial catalog = SubscriberData**.
8. Click Credentials Stored Securely On The Report Server.
9. Enter your username in the User Name box in the ***domain\username*** format. Then enter your password in the Password text box.
10. Check the Use As Windows Credentials When Connecting To The Data Source box.

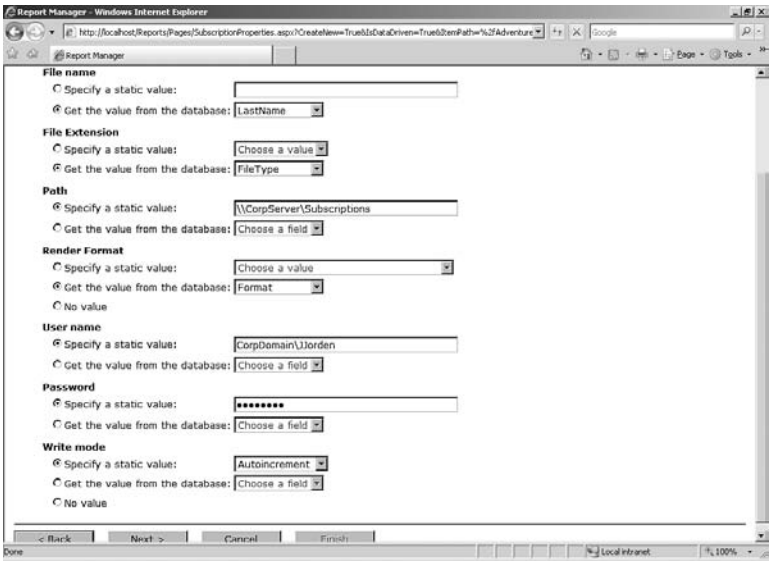


11. Click Next.
12. Enter this query in the query pane:

```
SELECT SubscriptionID, EmployeeID, LastName,
FileType, Format FROM UserInfo
```



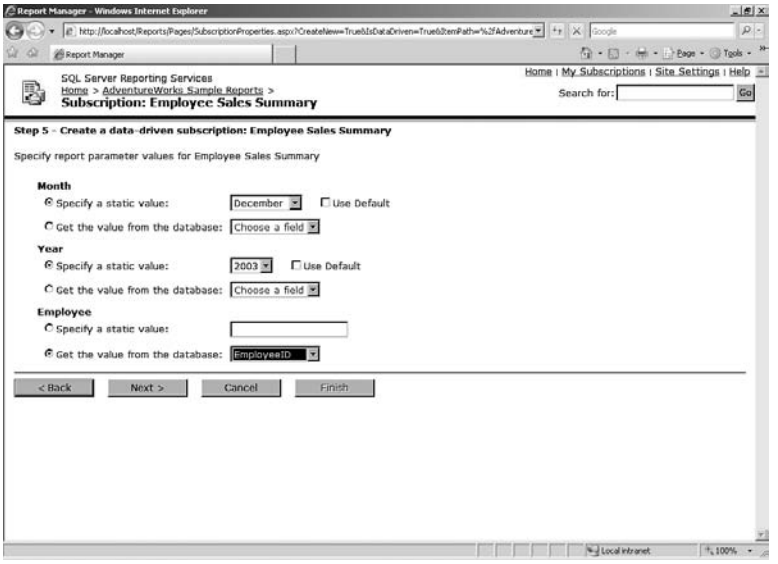
13. Click the Validate button to make sure your query is correct. If it is, click Next.
14. Under File name select the Get The Value From The Database option and select LastName from the list of fields.
15. Under File Extension, select the Get The Value From The Database option and select FileType from the list of fields.
16. Under Path, select the Specify A Static Value option and enter the UNC path to the subscriptions share you created earlier in this chapter. It should look like this: *servername***Subscriptions**.
17. Under Render Format, select the Get The Value From The Database option and select Format from the list of fields.
18. Under User Name, select the Specify A Static Value option and enter your username in the text box. It should look like ***domain username***.
19. Under Password, select the Specify A Static Value option and enter your password in the text box.
20. Under Write mode, select the Specify A Static Value option and select AutoIncrement. The Write mode setting defines whether existing reports will be overwritten with new versions when the subscription runs. Setting the value to AutoIncrement leaves old reports intact and creates a new report with an automatically generated name.



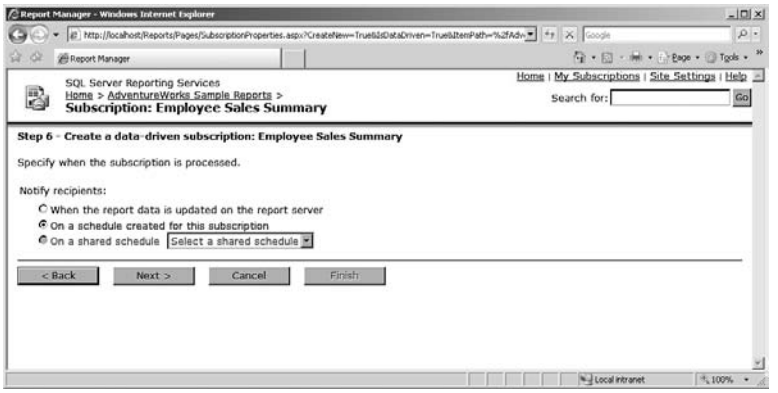
21. Click Next.

22. Use the default parameter values for Month and Year.

23. Under Employee, select the Get The Value From The Database option and select EmployeeID from the list of fields.



24. Click Next.
25. Select the On A Schedule Created For This Subscription option.



26. Click Next.
27. Select the Once option.
28. In the Start Time, enter a time that is 10 minutes from the current time.
29. Click Finish.

When the subscription runs, it will create three reports, one for each record in the UserInfo table. Each report will contain unique data and have a unique name. Now, let's see the real power of data-driven subscriptions by adding a user and changing the delivery extension of one of the existing users:

1. Start Management Studio, and open a connection to a Database Engine.
2. Click the New Query button on the toolbar.
3. Enter this query to add a new record to the table:

```
USE SubscriberData
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('5', '132', 'Uddin', '1', 'IMAGE')
GO
```

4. Click the Execute button on the toolbar.
5. Click the New Query button on the toolbar.
6. Enter this query to change the Valdez report from the Image format to PDF:

```
USE SubscriberData
UPDATE [dbo].[UserInfo] SET Format = 'PDF'
WHERE SubscriptionID = '4'
GO
```

7. Click the Execute button on the toolbar.
8. Open Report Manager and click the My Subscriptions link.
9. Click the Edit link next to the Employee Sales subscription.
10. Click the Select Schedule button and enter a time that is 10 minutes from the current time for the Start Time.
11. Click OK to create the schedule.
12. Click OK again to apply the changes to the report.

When the report runs again in 10 minutes, you will see a report for the new record you added to the database and the Valdez report will be in a PDF format instead of an image format.

Real World Scenario

CASE STUDY: CHOCOLATY GOODNESS CANDIES

When I worked for them, CGC was a mid-sized company with a sales staff of about 50 people. We created a database for the department that held, among other things, information about appointments for each salesperson on staff. To make this information easy to access, we created a report that accepted a salesperson's name and displayed their upcoming appointments.

The sales staff was not always at the office, though, so they could not generate the report manually when they needed it. Realizing that each salesperson required his or her own report, we created a data-driven subscription that created a unique report for each salesperson. We decided to use the e-mail delivery provider so that the salespeople could get the report right in their e-mail inbox, even when they were on the road.

SharePoint Integrated

Subscriptions work a little differently if you have SQL Server 2005 Service Pack 2 installed and you are running in SharePoint integrated mode.

One important change that affects subscriptions is the way that reports are stored in SharePoint integrated mode. Reports are stored in SharePoint document libraries, which are like folders in Reporting Services, but with more features.

You can create subscriptions based on reports stored in SharePoint document libraries, and you can save the reports generated in another document library in the same SharePoint site. You cannot save the report in a document library on another server or in another site in the same site collection, but you can save the report to a shared folder on the network or e-mail the file to a recipient.

There is another important difference you need to know about. When you create a subscription that stores a report in a document library, it is stored as a static file. This means that reports created by a subscription and stored in a document library do not have the same dynamic features that a report stored in a Reporting Services folder might have. For example, you will not be able to drill down into subreports. Also, you cannot rename the file to give it an `.rdl` extension and expect it to work dynamically, because the file is not based on the report definition language.

SharePoint document libraries support automatic versioning, so if you save a document to a library with versioning enabled it will create a new version of the document rather than

overwriting the original. If you have a subscription save a report to a document library that has versioning enabled, the library will create a new major version of the report every time the subscription runs. To create a subscription that delivers the report to a SharePoint document library:

1. Open the SharePoint library that contains the report.
2. Hover the mouse over the report, click the down arrow that appears next to the report, and click Manage Subscriptions.
3. Click Add Subscription.
4. Select SharePoint Document Library in Delivery type.
5. In Document Library, select a library within the same site.
6. In File Options, specify the filename and title for the document.
7. Choose a format in Output Format. The default is Web Archive (MHTML) because it generates a self-contained HTML file.
8. In Overwrite Options, specify an option that determines whether subsequent deliveries overwrite a file.
9. In Delivery Event, specify a schedule or event that causes the subscription to run.
10. If you are creating a subscription for a parameterized report, specify the values under Parameters.

Managing Subscriptions

After your users start creating subscriptions, someone will need to keep an eye on them to make sure they are running as expected. The user who created the subscription will need to perform certain tasks, and the administrator must take care of other tasks.

Individual users can monitor the status of their own subscriptions in Report Manager on the My Subscriptions page or the Subscriptions tab of any report they subscribe to. Both of these pages feature columns that display information about the status of the subscription. Table 9.1 lists the status indicators and what they mean.

If a there is a problem generating or delivering a report, the system administrator can check the `reportserverservice_`*`timestamp`*`.log` files. Nothing is recorded in the log files when a report is successfully generated and delivered, so the log files are only useful if there is an error. The log files are stored in the *`x`*`:\Microsoft SQL Server\`*`Server Instance`*`\Reporting Services\Logfiles` directory.

If the subscription becomes inactive, you will need to either delete and re-create it or fix the underlying problem that is preventing the subscription from generating reports. These are some of the most common conditions that cause a subscription to become inactive:

- Removing the delivery extension that the subscription uses to deliver reports
- Changing the credential settings of the data source to integrated or prompted values
- Changing a parameter name or data type in the report that is being subscribed to
- Changing the execution mode of a report, e.g., so that it runs on a schedule instead of a snapshot event

TABLE 9.1: Subscription Status Messages

STATUS MESSAGE	DESCRIPTION
New Subscription	The subscription has just been created and has not yet run.
Done: *x* processed of *x*; *x* errors	The number of errors and successfully generated reports by a data-driven subscription.
Inactive	The subscription cannot be processed.
Failure sending mail: the transport failed to connect to the server	The Report Server could not connect to the SMTP server to deliver reports via e-mail.
File *filename* was written to *path*	The report was successfully delivered to a file share.
An unknown error occurred when writing a file	The report was not delivered to a file share for an unknown reason.
Failure connecting to destination folder *path*. Verify the destination folder or file share exists.	The subscription could not find the file share that was specified for report delivery.
The file *filename* could not be written to *path*. Attempting to retry.	A file being delivered to a file share could not be updated with a newer version.
Failure writing *filename*: *message*	The delivery to the file share did not succeed for the reason stated in the message.

Preloading the Report Cache

Reporting Services can cache reports to accelerate access times. The first time a user accesses a report that has been configured for caching, the Report Server processes the report normally, returns the result to the user, and saves a copy of the processed report in the Reporting Services database. When users make subsequent requests for that report, Reporting Services returns the preprocessed report from the cache. When the cached report expires, the cycle starts over again.

Reporting Services caches a single instance of a report based on the report name, unless the report accepts parameters. If the report accepts parameters, Reporting Services caches a copy of each variation of the report that is accessed. For example, suppose you have a sales report that accepts a state as a variable. When one of the users accesses the report using the CA state abbreviation, Reporting Services caches the report with the `CA` variable. Another user then accesses the sales report using the MN state abbreviation and Reporting Services caches a second copy of the report. Subsequent users that use the `CA` or `MN` values will receive a cached report.

The primary difference between the cached and noncached reports is the response time. Reports are removed from the cache based on the expiration policy you set when you create the report.

Notice in the sequence of events that when a user accesses a report for the first time, it is processed normally. You can create a subscription to preload the cache so that the report is returned from the cache the first time it is accessed. Going one step further, you can create a data-driven subscription to preload the cache with multiple versions of a report that accepts parameters.

The secret to preloading the cache using a subscription is the Null Delivery Provider. This special provider targets the Report Server database as the delivery destination for the report and is therefore only useful for preloading the cache.

Next, you will configure a report for caching and create a subscription to preload the report into the cache. First, though, you should execute the report in a noncached mode so you can see the difference. Only one step is involved in executing the noncached report:

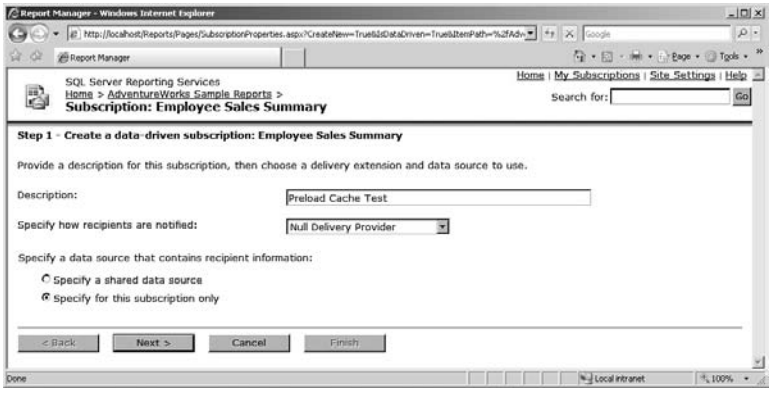
1. In Report Manager, on the Home page, click the AdventureWorks Sample Reports link, open the Employee Sales report, and note how long it takes to return results.

Now you need to configure the report to run from the cache:

1. On the Home page in Report Manager, click the AdventureWorks Sample Reports link and open the Employee Sales report.
2. On the Properties tab, click the Execution link.
3. Select the option labeled Cache a Temporary Copy of the Report. Expire Copy of Report After a Number of Minutes.
4. Click Apply.

At this point, you are ready to create a subscription to preload the cache:

1. On the Home page in Report Manager, click the AdventureWorks Sample Reports link and open the Company Sales report.
2. Click the Subscriptions tab, and then click New Data-Driven Subscription.
3. Type a description for the subscription if you like; this step is optional.
4. Select Null Delivery Provider from the delivery method drop-down list.
5. Make sure that Specify For This Subscription Only is selected for the recipient information data source and click Next.



6. Select Microsoft SQL Server as the data source type from the Connection Type drop-down list.
7. In Connection String, type the following connection string: **data source = localhost; initial catalog = SubscriberData**.

8. Click Credentials Stored Securely On The Report Server.
9. Next, to connect to the subscriber data source, enter your username in the User Name box in the ***domain\username*** format. Then enter your password in the Password text box.
10. Check the Use As Windows Credentials When Connecting To The Data Source box.
11. Click Next.
12. Enter this query in the query pane:

```
SELECT SubscriptionID, EmployeeID, LastName,
FileType,  Format  FROM  UserInfo
```

13. Click the Validate button to make sure your query is correct. If it is, click Next.
14. On the Step 4 page, click Next.
15. On the Step 5 page, click Next.
16. Select the On A Schedule Created For This Subscription option.
17. Click Next.
18. Select the Once option.
19. In the Start Time, enter a time that is 10 minutes from the current time.
20. Click Finish.

When the subscription executes, it will create the reports and store them in the cache. When people access these reports from the server, it will be much faster than if they were not cached.

The Bottom Line

Know how subscriptions are processed. The scheduling and delivery processor makes sure that reports are generated and delivered on schedule. So, to fully understand subscriptions, you need to know what this processor does when it runs a subscription.

> **Master It** What are the steps that the scheduling and delivery processor takes to process a subscription?

Create standard subscriptions. Standard subscriptions create a single instance of a report for all users to view. The values that the subscription uses to create the report cannot be changed at runtime.

> **Master It** You have created a report that displays company sales figures. The sales manager would like to subscribe to the report. You do not need to modify the report parameters at runtime, so you decide to recommend a standard subscription that uses the file-share delivery method. How would you have the sales manager create the subscription?

Create data-driven subscriptions. Data-driven subscriptions can create multiple reports in multiple formats, each having a unique set of data. They do this by retrieving subscription settings from a data source.

> **Master It** You have created a report that displays company sales figures. Several of the employees in the sales department would like to get this information on a regular basis,

but they need it in different formats. You realize that the only way to satisfy this requirement is with a data-driven subscription. How would you create the subscription?

Know how to manage subscriptions. After your users start creating subscriptions, they will come to rely on them to perform their day-to-day duties. You will have to manage those subscriptions to make sure that they are running as expected. The users will be able to check the status of their own subscriptions in Report Manager, but you must fix any issues that may arise.

Master It One of your users is complaining that they are not receiving a report from a subscription on a regular basis. The subscription had been working fine until just a few days earlier. On investigation, you find that the subscription has become inactive. What should you check?

Know how to use subscriptions to preload the report cache. Reporting Services can cache reports to accelerate access times. Reporting Services processes the report normally the first time it is accessed; subsequent requests are filled from the cache. You can preload the cache so that all user requests are filled from the cache by creating a subscription that targets the Null Delivery Provider as the report destination.

Master It The users that have subscribed to the Company Sales report are complaining that it is too slow. You decide to have the report cached to speed up access, but you do not want the first user who accesses the report to experience lag time. How can you preload the report into the cache?

Chapter 10

Maintenance and Administration

If you own a car, you know that you need to perform regular maintenance on it. You have to get the oil changed regularly, make sure the tires are properly inflated, get the brake pads replaced when they wear out, schedule regular tune-ups, and the list goes on. If you do not get the required regular maintenance on your car, it will slowly start to wear down and eventually will stop running altogether.

Much like your car, Reporting Services requires regular maintenance. Without that maintenance, your reporting system will start to slow down and eventually will stop working altogether. So it is important that you administer and maintain your reporting system once it is in production.

A number of components require your attention, one of which is the main Report Server database. Reporting Services uses two databases to accomplish its tasks; a temporary database, ReportServerTempdb, and the main database, ReportServer. The ReportServerTempdb database does not contain any persistent data, so it requires little maintenance. On the other hand, the ReportServer database requires regular attention because it houses persistent data, such as:

- Reports, linked reports, shared data sources, report models, folders, resources, and all of their associated properties and security settings
- Subscription and schedule definitions
- Report snapshots (including query results)
- Report history
- System properties and system-level security settings
- Report execution log data
- Symmetric keys, encrypted connection, and credentials for report data sources

Notice that one of the items stored in the ReportServer database is encrypted information. Reporting Services requires keys to encrypt that information, and you will need to manage those keys. Specifically, you should know how to back up and restore the encryption keys, and what to do if the keys are irretrievably lost.

Because Reporting Services stores content, you also have to play the role of content manager. In this role you:

- Set report snapshot sizes and limits
- Configure report history properties
- Manage My Reports
- Manage resources
- Create, modify, and delete folders

You must also manage the settings that Reporting Services uses to process reports. For instance, you may have to configure report caching for some reports to speed up report access. You may also need to pause subscription and report execution, as well as manage running processes.

Of course, nothing is trouble free, so you can expect to run into the occasional snag. To help overcome these problems, you should know where Reporting Services stores its log files and how to read those log files. In this chapter, I'll show you some techniques for solving some of the most common problems that arise with Reporting Services.

In this chapter you will learn how to:

- Manage the Windows and web services
- Back up and restore the Report Server database
- Manage encryption keys
- Manage report processing and content
- Troubleshoot common problems

Managing the Report Server

Reporting Services consists of several components that require regular maintenance. You should back up the Report Server database regularly so that you do not lose data in the event of a disaster. You also need to back up the encryption keys for disaster protection as well. The Windows and web services that power Reporting Services require some maintenance too.

There are a few requirements that you need to meet to administer Reporting Services:

- You must be a local administrator on the server running Reporting Services.
- You have to be a database administrator on the SQL Server database server that houses the Reporting Services databases.
- If you install Reporting Services on a domain controller, you must be a domain admin.

NOTE

For the remainder of this chapter, I assume that you are a local administrator and database administrator.

If you need to add a Reporting Services administrator but you do not want to add that individual to the local administrators group, you can assign that person to the System Administrator role (the local administrators group is already assigned to the role). Table 10.1 describes the tasks included in the System Administrator role.

In this section, you will see how to administer each of the components that make up the Report Server, starting with the Windows and web services.

Administering the Services

Reporting Services runs as two separate services: a Windows service and a web service. Each service provides different functionality:

- Windows service
  - Performs initialization
  - Provides reversible encryption

- Performs database maintenance
- Enables scheduling and delivery

- Web service
  - Processes ad hoc reports
  - Provides the programming interface that developers use to integrate applications with Reporting Services

TABLE 10.1: System Administrator Role Tasks

TASK	DESCRIPTION
Execute Report Definitions	Execute a report definition without publishing it to a server. You can do this using Report Builder.
Manage Jobs	View and cancel running jobs.
Manage Report Server Properties	Rename Report Manager, enable My Reports, and set report history defaults.
Manage Roles	Create, view, modify, and delete roles.
Manage shared schedules	Create, view, modify, and delete shared schedules.
Manage Report Server Security	View and modify systemwide role assignments.

In most circumstances, you will run both of these services on the same server. There are a few situations where it may be useful to disable one of the services:

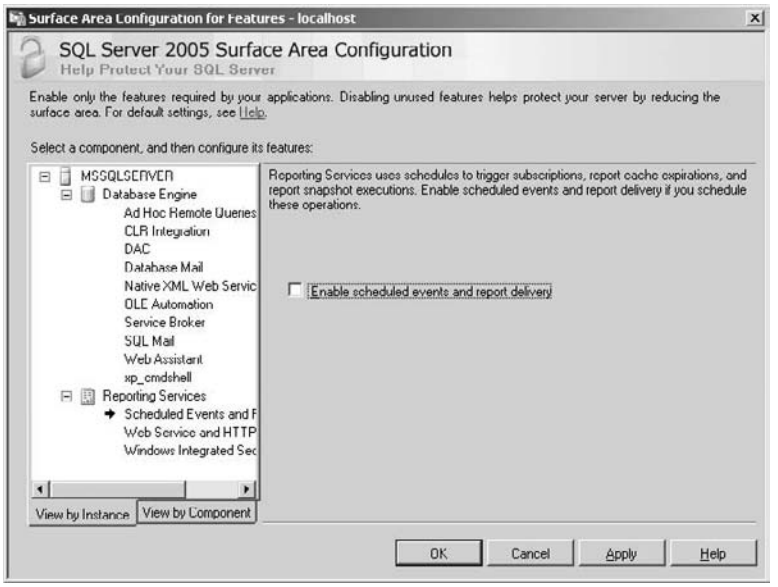
- If you have a large organization that uses Reporting Services heavily, you can separate the services on different machines.
- If you do not want the server to generate scheduled reports, you can disable the Windows service.
- If you do not want users to create ad hoc reports, you can disable the web service.

NOTE

You cannot turn off the Windows service entirely; you can only disable scheduled processing and delivery.

You use the SQL Server Surface Area Configuration tool to enable or disable the services. Here are the steps to disable the scheduling and delivery functions of the Windows service (do not perform these steps for this chapter; this is only a reference):

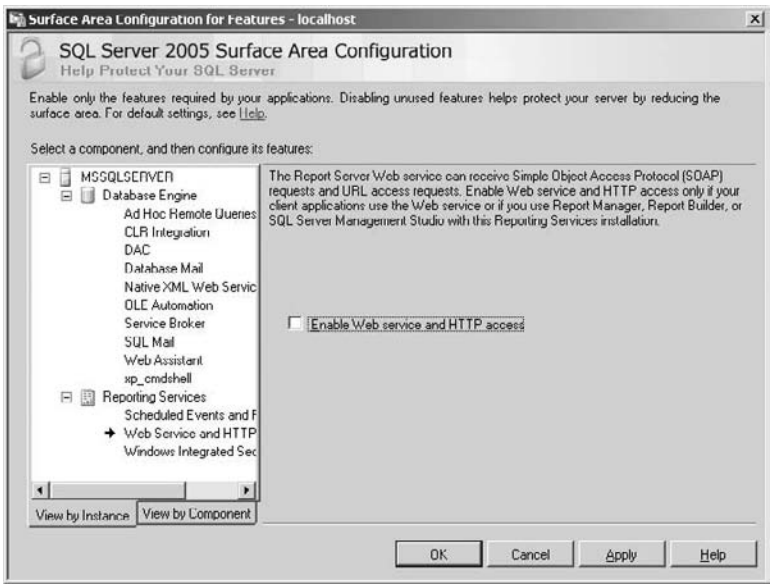
1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ SQL Server Surface Area Configuration.
2. Click Surface Area Configuration For Features.
3. Expand Reporting Services and click on Scheduled Events And Report Delivery.
4. Deselect the Enable Scheduled Events And Report Delivery check box.



5. Click Apply.

To disable the web service:

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ SQL Server Surface Area Configuration.
2. Click Surface Area Configuration For Features.
3. Expand Reporting Services and click on Web Service And HTTP Access.
4. Deselect the Enable Web Service And HTTP Access check box.



5. Click Apply.

The Windows service also relies heavily on the SQL Server Agent service for scheduled operations, so make sure they are configured to work together. If the Reporting Services Windows service was configured to run using a domain service account, the SQL Server Agent service will need to run under a domain service account as well. This is because the Reporting Services Windows service creates jobs for the SQL Server Agent service to run. When Reporting Services creates these jobs, it creates them using its domain account. The SQL Server Agent service requires permission to access job information for jobs created by domain accounts, which it will only have if it is configured to run using a domain account. So if the Reporting Services Windows service is configured to run using a domain account and the SQL Server Agent service is not, scheduled operations will fail. So you should use a different account for each service, and both accounts should belong to the same domain.

Both of these services can also be stopped if necessary. Stopping the Windows service does not prevent users from accessing reports from Report Manager or from the SOAP interface. The following functions are no longer available when you stop the Windows service:

- All report processing that is triggered by a scheduled operation is stopped.
- All encryption operations, like initializing servers and accessing encrypted credentials and connection strings, are stopped.
- The Remote Procedure Call endpoint that is used by most of the configuration tools is closed.
- Database maintenance operations are halted.

If you disable the web service using the SQL Server Surface Area Configuration tool, you may want to stop the web service. You can stop the service by stopping the application pool in IIS Manager.

Performing Backups

A backup is a copy of your data that is stored somewhere other than the hard drive of your computer, usually on some type of magnetic tape. Backups are a vital part of your recovery strategy for several reasons.

The first reason for keeping a backup is hardware failure. Computer hardware has a Mean Time Between Failures (MTBF) that is measured in hours. This means that every 4,000 hours or so, a piece of hardware is going to fail, and there is little you can do about it. True, you could implement fault tolerance by providing duplicate hardware, but that is not a complete guarantee against data loss. So if you don't want to lose your data when a hard disk goes bad, it is best to back up.

Another reason is natural disaster. No matter how much redundant hardware you have in place, it is not likely to survive the wrath of a tornado, hurricane, earthquake, flood, or fire. To thwart the wrath of the elements, you need to back up your data and store the backup medium in a secure place.

A third reason is malicious destruction or updating of sensitive data by disgruntled employees or hackers who breach your security. This is the worst kind of data loss, and the only way to recover from it is by having a viable backup.

NOTE

This is not intended to be a detailed discussion of backup and restore. For more information on the subject, see *Mastering SQL Server 2005* (Sybex, 2006).

Now that you have some good reasons to back up your data, you need to know how to do it. The type of backup you perform is governed in part by the recovery model you select for the database. There are three recovery models:

Simple With this recovery model, every time SQL Server transfers data from the transaction log to the database, the information is deleted from the transaction log. Databases that use this model do not require a transaction log backup. This is the recommended model for the ReportServerTempdb database, because it is not used to store persistent data.

Bulk-Logged This is a special recovery model that is used only during bulk import operations. You should not use this model for any of the Reporting Services databases.

Full With this model, all information is recorded to the transaction log, which is not cleared until a transaction log backup is performed. This model gives you the ability to recover a database up to the minute of failure. This is the recommended model for the ReportServer database.

Backup Types

There are three types of backup that you can incorporate into your backup strategy:

Full Backups Just as the name implies, a full backup is a backup of the entire database. It backs up the database files, the locations of those files, and portions of the transaction log (from the Log Sequence Number [LSN] recorded at the start of the backup to the LSN at the end of the backup). This is the first type of backup you need to perform in any backup strategy because all the other backup types depend on the existence of a full backup. This means that you cannot perform a differential or transaction log backup if you have never performed a full backup.

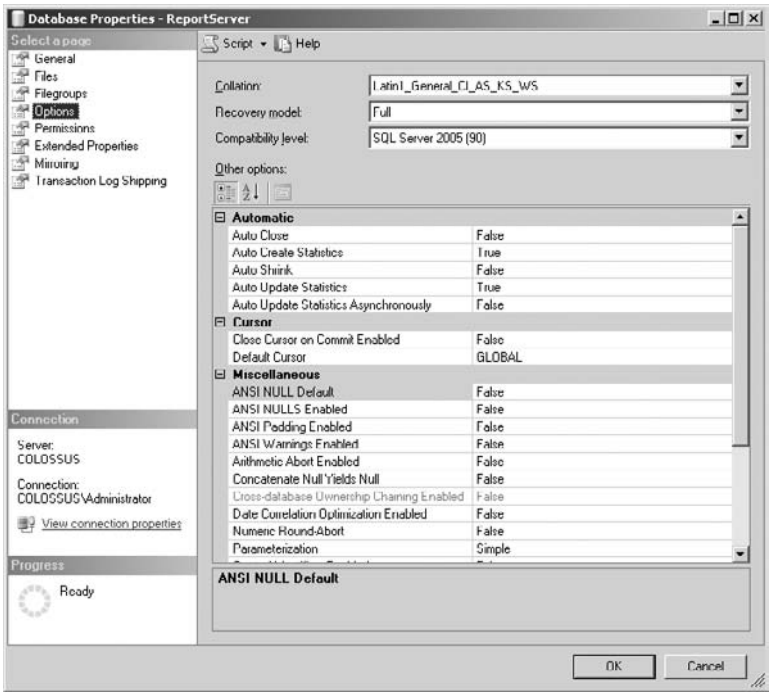
Differential Backups Differential backups are designed to record all the changes made to a database since the last full backup was performed. This means that if you perform a full backup on Monday and a differential backup on Tuesday, the differential would record all the changes to the database since the full backup on Monday. Another differential backup on Wednesday would record all the changes made since the full backup on Monday. The differential backup gets a little bigger each time it is performed, but it is still a great deal smaller than the full backup, which makes a differential backup faster than a full backup.

Transaction Log Backups Although they still rely on the existence of a full backup, transaction log backups don't actually back up the database itself. This type of backup only records sections of the transaction log, specifically since the last transaction log backup. When a database is configured to use the Full or Bulk-Logged recovery model, a transaction log backup is the only type of backup that will clear old transactions out of the transaction log. Therefore, if you were to perform only full and differential backups on most production databases, the transaction log would eventually fill to 100 percent capacity, and your users would be locked out of the database.

The ReportServerTempdb database does not hold persistent data, so you do not need to back up the data in the database. However, you do need the table structure in the event of a disaster, so you should back it up right after installation and upgrades.

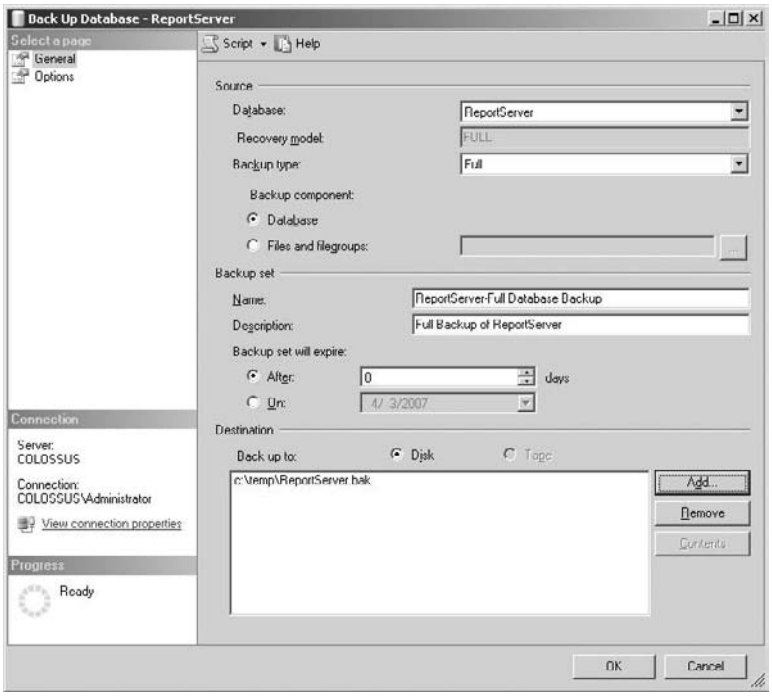
Back up the ReportServer database on a regular basis. You may do this by following a database maintenance plan, or use a third-party backup software package. For practice, though, let's perform a full backup on the ReportServer database:

1. Open SQL Server Management Studio, connect to the Database Engine, expand your server, and then expand Databases.
2. Right-click ReportServer and select Properties.
3. On the Options page, verify that the Recovery model is set to Full.



4. Click OK.
5. Right-click ReportServer under Databases, point to Tasks, and click Back Up.
6. In the Backup dialog box, make sure that ReportServer is the selected database to back up and that the Backup type is Full.
7. Leave the default name in the Name box, and in the Description box, type **Full Backup of ReportServer**.
8. Under Destination, there may already be a disk device listed. If so, select the device and click Remove.
9. Under Destination, click Add.
10. In the Select Backup Destination box, select the File Name option and enter **c:\temp\ReportServer.bak** in the text box; then click OK.

11. Check that you now have a backup device listed under Destination.



12. Click OK to start the backup and click OK again when it is complete.

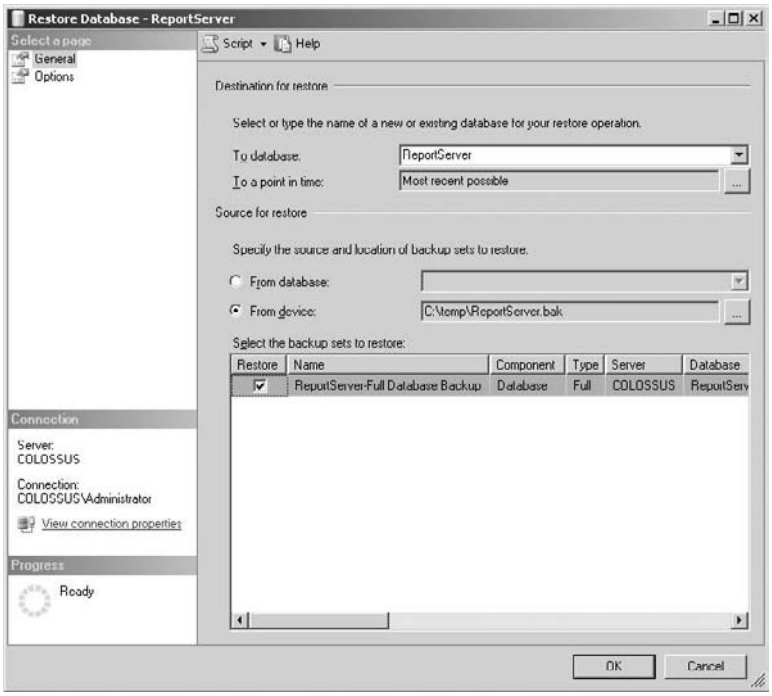
It is also important that you know how to restore the Reporting Services databases in the event of a crash. To make this more meaningful, let's make one of the databases suspect so that you can see exactly what SQL Server does to restore it. Specifically, let's take ReportServer down:

1. Open the SQL Server Computer Manager from the Start menu.
2. Expand Services and select SQL Server.
3. Right-click SQL Server in the right pane and click Stop. You will be asked whether you wish to stop the SQLServerAgent service as well; click Yes. This is necessary because the data files are open as long as SQL Server is running.
4. Find the filename `ReportServer.mdf` (usually in `C:\Program Files\Microsoft SQL Server\MSSQL\Data\`).
5. Rename the file **ReportServer.old**.
6. Find the file named `ReportServer_log.ldf` and rename it **ReportServer_log.old**.
7. From the Computer Manager, restart the SQL Agent and SQL Server services.

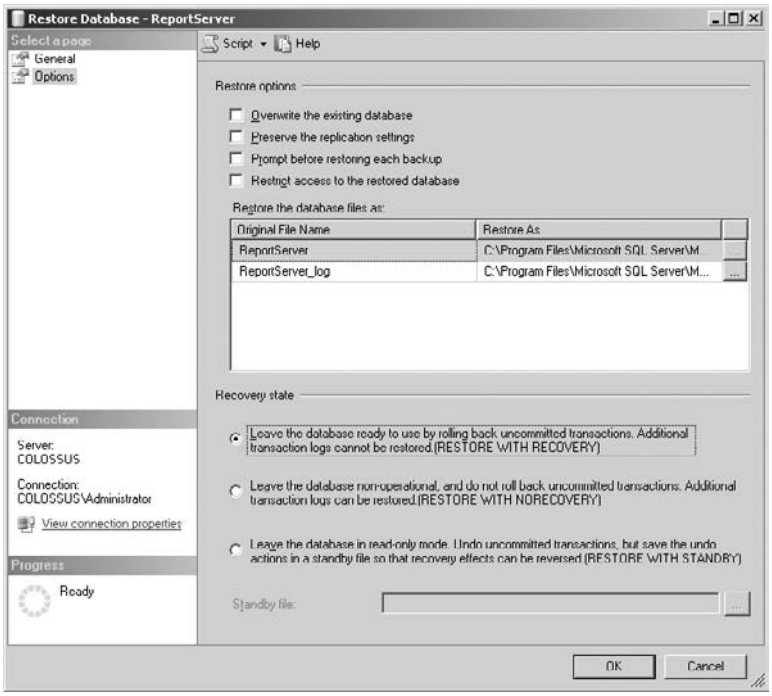
8. Open Management Studio and expand databases under your server name. You should not be able to expand the ReportServer database.

Now that the ReportServer database is suspect, you can restore it:

1. Right-click Databases and select Restore Database.
2. In the Restore Database dialog box, select ReportServer from the To Database drop-down list box.
3. Under Source For Restore, select From Device and click the ellipsis button next to the text box to select a device.
4. In the Specify Backup dialog box, select File from the Backup Media drop-down list box and click Add.
5. Browse for the `c:\temp\ReportServer.bak` file you created earlier and click OK.
6. Click OK to close the Specify Backup dialog box.
7. Under Select The Backup Sets To Restore, check the full backup.



8. On the Options page, make sure the RESTORE WITH RECOVERY option is selected because you have no more backups to restore.



9. Click OK to begin the restore process.
10. In Management Studio, right-click Databases and click Refresh.
11. Expand Databases and you should see ReportServer back to normal.

Remember that Reporting Services comprises a number of different components, so there is more to back up than just the databases:

- Back up the encryption keys right after installation and any time you change the service account or rename the computer.
- Back up the Reporting Services websites right after installation.
- Back up the configuration files after installation and any time you make a change to them. The files include:
  - `RSReportServer.config`
  - `RSWebApplication.config`
  - `RSSvrPolicy.config`
  - `RSMgrPolicy.config`
  - `ReportingServicesService.exe.config`
  - `Web.config` for both the Report Server and Report Manager `ASP.NET` applications
  - `Machine.config` for `ASP.NET`

- Back up the files created in Report Designer and Model Designer, including:
  - Report definition files (`.rdl`)
  - Report model files (`.smdl`)
  - Shared data source files (`.rds`)
  - Data view files (`.dv`)
  - Data source files (`.ds`)
  - Report server project files (`.rptproj`)
  - Report solution files (`.sln`)
  - Script files (`.rss`)
  - Custom assembly files (`.dll`)

Managing Encryption Keys

Before you can put a Report Server to work after installation, it must be initialized. The Report Server is initialized when you start the Report Server Windows service for the first time, when you join a server to an existing deployment, or when you manually re-create keys during a recovery. When a server is initialized, it obtains an encryption key that it can use to store and retrieve encrypted information from the Reporting Services databases. The initialization process looks like this:

1. The first time you start the Report Server Windows service, it reads the `RSReportServer.config` file to get the installation identifier and database connection information.
2. The Report Server Windows service requests a public key from Cryptographic Services. In response, Windows creates a private and a public key and sends only the public key to the Report Server Windows service.
3. The Report Server Windows service stores the installation identifier and public key values in the Report Server database.
4. The Report Server Windows service calls into Cryptographic Services a second time to request a symmetric key.
5. The Report Server Windows service uses the public key to encrypt the symmetric key.
6. The Report Server Windows service adds the encrypted symmetric key to the public key and installation identifier values that were stored in the Report Server database in step 3.
7. Once the symmetric key is stored, the Report Server is considered initialized and available to use.

Once initialized, the Report Server can work with encrypted information. The following is a list of encrypted values that Reporting Services may need to work with:

- The information used to connect to the Report Server database is encrypted and stored in the `RSReportServer.config` file. This is encrypted using the machine-level key that is available to all users, not the symmetric key.
- Stored credentials used by data sources are encrypted using the symmetric key.

- When a report is configured not to use credentials and requires external resources, Reporting Services uses a special unattended user account to access those resources. The unattended user account information is encrypted and stored in the `RSReportServer.config` file.
- The symmetric key itself is encrypted.

The public and private keys created by Windows Cryptographic Services are managed by the operating system. Managing the symmetric key is your responsibility. To start, you need to back up the encryption keys, which you can do using the `rskeymgmt.exe` command-line tool. Let's do that here:

1. Put a floppy disk in the floppy drive if you are planning to store the file separately from the Report Server. This is optional, but recommended.
2. Run `rskeymgmt.exe` on the Report Server, using the `-e` argument to extract the key, the `-f` argument to specify a location for the file on disk, and the `-p` option to protect the key with a password. Here is the command:

```
rskeymgmt -e -f a: \RSKey.snk -p password
```

Here is the process to restore the encryption keys:

1. Insert the disk with the saved key in the floppy drive.
2. Run `rskeymgmt.exe` on the Report Server, using the `-a` argument to restore the key, the `-f` argument to specify a location for the file on disk, and the `-p` option to open the key with a password. Here is the command:

```
rskeymgmt -a -f a: \RSKey.snk -p password
```

While it is not a normal administrative task, it is possible to delete and re-create the symmetric key. You may need to do this if you are concerned that your symmetric key has been compromised in some way, or if you work for an organization that requires stringent security measures.

The Report Server must be in a working state to delete and re-create the key. Because all encrypted data will be re-encrypted using the new symmetric key, users will not be able to use the server while re-encryption is under way, so you should try to do this off-hours. Here is the process for deleteing and re-creating the encryption key (do not do this now; this is only a reference):

1. First, you must disable the Report Server web service, so start the SQL Server Surface Area Configuration tool and click Surface Area Configuration For Features.
2. Select Reporting Services.
3. For Web Service And HTTP Access, clear the check box Enable Web Service And HTTP Access.
4. Run `rskeymgmt.exe` on the Report Server, using only the `-s` argument to reset the key. Here is the command:

```
rskeymgmt -s
```

5. Restart the Windows service and enable the web service.

Managing Processing and Content

Along with managing the Report Server, you need to manage the content that is stored on the server. This content includes reports, models, folders, resources, and shared data sources. Users must be assigned to the Content Manager role to manage content. Table 10.2 describes the tasks assigned to this role. Local administrators are already members of the Content Manager role.

TABLE 10.2: Content Manager Role Tasks

TASK	DESCRIPTION
Consume Reports	Read report definitions.
Create Linked Reports	Create a linked report based on a nonlinked report.
Manage All Subscriptions	View, modify, and delete subscriptions for any report, regardless of the owner, and create data-driven subscriptions.
Manage Data Sources	Create and delete shared data sources, and view and modify data source properties.
Manage Folders	Create, view, modify, and delete folders.
Manage Models	Create, view, and delete models, and modify model properties.
Manage Individual Subscriptions	Create, view, modify, and delete user-owned subscriptions.
Manage Report History	Create, view, and delete report history; view report history properties; and view and modify settings that affect snapshot history and caching.
Manage Reports	Add and delete reports; modify report parameters; view and modify report properties, data sources, and report definitions; and set report-level security policies.
Manage Resources	Create, modify, and delete resources and modify resource properties.
Set Security Policies for Items	Set security policies for reports, linked reports, folders, resources, and data sources.
View Data Sources	View shared data sources.
View Reports	Run reports and view report properties.
View Models	View models, use models as data sources, and query models.
View Resources	View resources and resource properties.
View Folders	View folder contents and navigate the hierarchy.

Report Settings

Each report that is published to the Report Server has properties that you can set. The Properties tab for each report is split up into six sections: General, Parameters, Data Sources, Execution, History, and Security.

On the General page, you can change the name and description of the report, hide the report in the list so users cannot see it (this is useful for subreports that must be called from a parent report), and extract the Report Definition Language document for the report.

On the Parameters page, you can specify whether each parameter has a default value and set the default value. You can instruct Reporting Services to hide the parameter from the user (also useful for subreports that are called from a parent), specify whether the user should be prompted for a value for the parameter, and change the display text for the parameter.

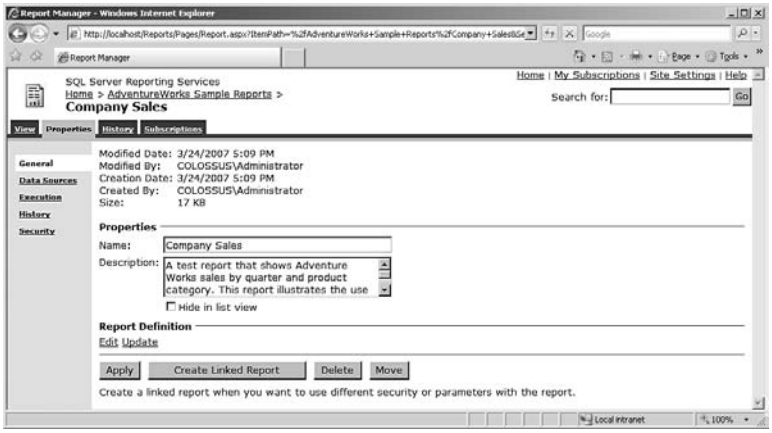
The Data Sources page lets you specify whether the report is using a shared or custom data source. If the report is using a custom data source, you can specify the connection settings for the custom data source.

On the Execution page, you can configure the report to be run from the report cache or to be run from a snapshot. You can also configure the report timeout from this page.

The History page allows you to specify whether the report history can be created manually, and whether report execution snapshots should be stored in history. You can also set a schedule to add snapshots to report history and set the number of snapshots to keep in history.

Finally, the Security page allows you to modify item-level security for the report. To practice, let's set the Description property of the Company Sales report here:

1. Start Report Manager.
2. In Report Manager, navigate to the Contents page, click the AdventureWorks Sample Reports link, and open the Company Sales report.
3. Click the Properties tab.
4. On the General page, type **A test report that shows** at the front of the existing description in the Description text box.



5. Click Apply.

The report history component deserves a little more detail. The report history is a collection of previously executed copies of a report, which allows you to maintain a record of a report

over time. The history is actually an extension of the report. If you move the report to another server, the history moves with it. If you modify the report or even delete its data source, the history is preserved. You can view the report history on the History page of the report's Properties tab.

My Reports

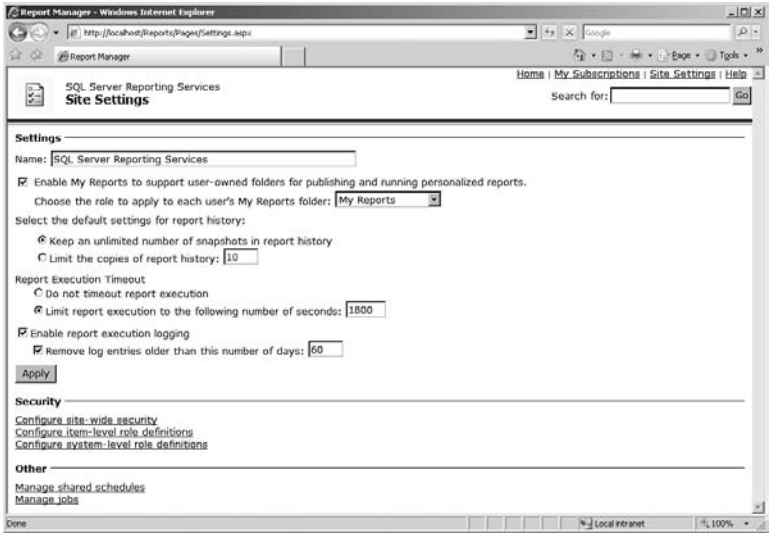
The My Reports feature creates a personal workspace where users can store and work with their own personal reports. You can enable or disable this feature for all users on the system, but there is no way to enable or disable it for select groups.

The most persuasive argument for enabling My Reports is ease of administration. With My Reports enabled, users can create reports in their own private workspace and control which users have access to those reports. Without My Reports enabled, you will need to control who has access to private reports, which can be an administrative headache because you will have to create and keep track of several custom roles and role assignments.

If you decide to enable My Reports, you must keep your resources in mind. Reporting Services will create a My Reports workspace for every user who clicks the My Reports link, even if they never store a private report in the workspace. This can be a waste of resources.

Follow these steps to enable My Reports:

1. Open Report Manager.
2. Click Site Settings.
3. Check the Enable My Reports To Support User-Owned Folders For Publishing And Running Personalized Reports check box.



4. Click Apply.

Managing Resources

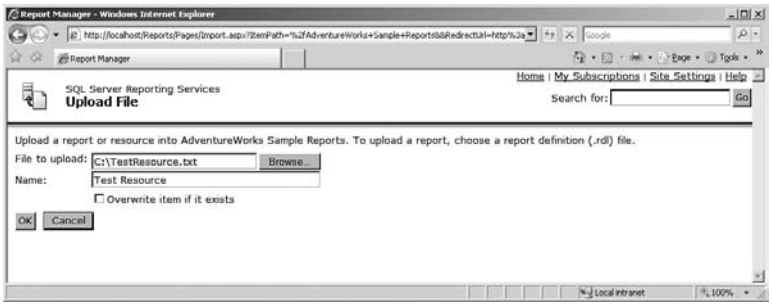
Reporting Services can store other types of files besides report files. These extra files are called resources, and they are handy for a few reasons. For example, you can upload an image file

(like a company logo) to a Report Server and embed the image in your report. If the company logo changes, all you need to do is update the resource file on the server to reflect the change in your report.

You can also create a Word document that explains what each report in a directory is for and how to use the parameters for the reports and upload that document to the Report Server. This can help cut down on help desk support calls from people who do not know where to find a specific report or what valid parameter values may be.

There are no restrictions on the file types that you can upload to the Report Server, and you can view and manage the resources just like any other object in the Report Server hierarchy. Let's upload a text file here to get a feel for how it works:

1. Create a text file on your C: drive named `TestResource.txt` and add some text to the file.
2. In Report Manager, navigate to the Home page and click the Sample AdventureWorks Reports link.
3. Click Upload File.
4. Click Browse and locate the `TestResource.txt` file you created in step 1.
5. Type **Test Resource** in the Name text box. The name can include spaces but it cannot include these reserved characters: ; ? : @ & = + , $ / * < > |.



6. Click OK.

Real World Scenario

Case Study: Chocolaty Goodness Candies

When I first installed Reporting Services at CGC, management wanted to organize the reports on the server according to team. So I created a folder for each cross-department team in the company. This worked out well for several months, until the company changed their management model and went from a team model to a department model. When that change took place, management asked me to reorganize the report folder hierarchy based on departments instead of teams.

From a technical standpoint, this was not a difficult process. The employees who had come to rely on Reporting Services were confused by the change, though, because they had trouble locating their reports. To assist the users in their search, I created a Word document that described each of the reports that were available, along with their old location and new location. I uploaded the file to the Report Server as a resource and the users were able to find their reports much easier.

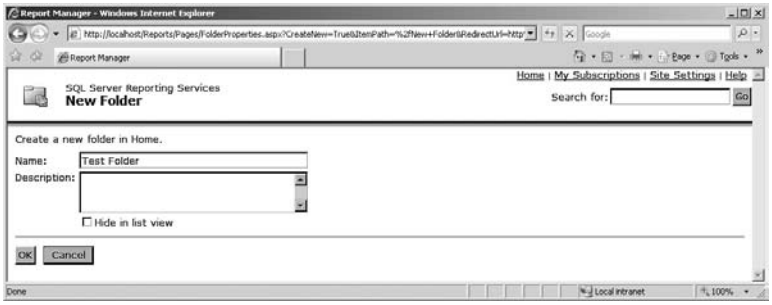
Managing Folders

You can organize your Report Server by creating folders. These look and act just like the folders you would create on a hard disk, but they are stored in the Report Server database. Folders are created in place, which means that you need to navigate to the parent folder where you want to create a subfolder before you create it; afterward, you can change the folder properties.

When you delete a folder, you also delete all of the contents of the folder. So, before you delete a folder, check its contents to make sure that you really want to delete everything inside. If you delete a folder that contains a report that is the basis for a linked report, the linked report will still exist but it will not function properly. If that is the case, make sure you delete the linked reports or move the base report to another folder before deleting the folder in question.

Follow these steps to create a folder in the Home directory:

1. Start Report Manager.
2. In Report Manager, select the Home folder and click New Folder.
3. The New Folder page opens.
4. On the New Folder page, type **Test Folder** in the Folder Name box. A folder name can include spaces, but cannot include reserved characters that are used for URL encoding: ; ? : @ & = + , $ / * < > |.



5. Click OK.

Managing a Running Process

There are two types of jobs that Reporting Services runs: system jobs and user jobs. System jobs include scheduled report processing, scheduled report history snapshots, and data-driven subscriptions. User jobs are anything that a user might initiate, such as an ad hoc report, a non-scheduled report history snapshot request, or the processing of a standard subscription. You can manage both of these types of processes.

You can find the status of a job in Report Manager on the Site Settings page under Manage Jobs (in the Site Settings page), or in Management Studio in the Jobs folder. By default, Reporting Services updates the status of these jobs every 60 seconds. It knows that a job is in progress if it is in the process of querying a database, processing a report, or rendering a report.

If a job is taking too long to complete, you may need to cancel the process because it is either stuck or just consuming too many resources. Here are the steps for canceling a running process:

1. Start Report Manager.
2. Click the Site Settings link at the top of the page.

3. Click the Manage Jobs link at the bottom of the page.
4. Check the box next to the job you want to cancel.
5. Click Cancel.

Troubleshooting

Nothing works perfectly, so you may run into trouble with Reporting Services from time to time. To troubleshoot these problems, you will need to know where to look for information, so in this section you will learn where the log files are and how to use them. I will also give you some pointers on troubleshooting some of the most common problems with Reporting Services.

Working with Log Files

When troubleshooting Reporting Services, check three log files: the Windows Application log, Reporting Services trace logs, and Report Server execution logs.

Windows Application Log

Like all Windows applications, Reporting Services writes events to the Windows Application log, so the Windows Application log is the first place to look when troubleshooting an issue. Reporting Services provides three event sources:

- Report Server (Report Server Windows service)
- Report Manager
- Scheduling and Delivery Processor

Reporting Services can log a number of different errors in the Application log. Those errors are listed in Table 10.3.

TABLE 10.3: Errors In the Windows Application Log

Event ID	Type	Category	Source	Description
106	Error	Scheduling	Report Server	SQL Server Agent must be running when you define a scheduled operation.
107	Error	Startup/ Shutdown	Report Server Scheduling and Delivery Processor	*Source* cannot connect to the Report Server database.
108	Error	Extension	Report Server Report Manager	*Source* cannot load a delivery, data processing, or rendering extension.
109	Information	Management	Report Server Report Manager	A configuration file has been modified.

TABLE 10.3: Errors In the Windows Application Log *(CONTINUED)*

EVENT ID	TYPE	CATEGORY	SOURCE	DESCRIPTION
110	Warning	Management	Report Server Report Manager	A setting in one of the configuration files has been modified so that it is no longer valid. A default value will be used instead.
111	Error	Logging	Report Server Report Manager	*Source* cannot create the trace log.
112	Warning	Security	Report Server	The Report Server has detected a possible denial-of-service attack.
113	Error	Logging	Report Server	The Report Server cannot create a performance counter.
114	Error	Startup/ Shutdown	Report Manager	Report Manager cannot connect to the Report Server service.
115	Warning	Scheduling	Scheduling and Delivery Processor	A scheduled task in the SQL Server Agent queue has been modified or deleted.
116	Error	Internal	Report Server Report Manager Scheduling and Delivery Processor	An internal error occurred.
117	Error	Startup/ Shutdown	Report Server	The Report Server database is an invalid version.
118	Warning	Logging	Report Server Report Manager	The trace log is not at the expected directory location; a new trace log will be created in the default directory.
119	Error	Activation	Report Server Scheduling and Delivery Processor	*Source* has not been granted access to the contents of the Report Server database.
120	Error	Activation	Report Server	The symmetric key cannot be decrypted. Most likely there has been a change to the account that the service runs as.
121	Error	Startup/ Shutdown	Report Server	Remote Procedure Call (RPC) Service failed to start.

(CONTINUED)

TABLE 10.3: Errors In the Windows Application Log *(CONTINUED)*

EVENT ID	TYPE	CATEGORY	SOURCE	DESCRIPTION
122	Warning	Delivery	Scheduling and Delivery Processor	Scheduling and Delivery Processor cannot connect to the SMTP server that is used for e-mail delivery.
123	Warning	Logging	Report Server Report Manager	The Report Server failed to write to the trace log.
124	Information	Activation	Report Server	The Report Server service has been initialized.
125	Information	Activation	Report Server	The key used for encrypting data was successfully extracted.
126	Information	Activation	Report Server	The key used for encrypting data was successfully applied.
127	Information	Activation	Report Server	Encrypted content was successfully removed from the Report Server database.
128	Error	Activation	Report Server	Reporting Services components from different editions cannot be used together.
129	Error	Management	Report Server Scheduling and Delivery Processor	An encrypted configuration file setting cannot be decrypted.
130	Error	Management	Report Server Scheduling and Delivery Processor	*Source* cannot find configuration file. Configuration files are required by the Report Server.
131	Error	Security	Report Server Scheduling and Delivery Processor	An encrypted user data value could not be decrypted.
132	Error	Security	Report Server	A failure occurred during encryption of user data. The value cannot be saved.
133	Error	Management	Report Server Report Manager Scheduling and Delivery Processor	A configuration file failed to load. This error may occur if the XML file is not valid.
134	Error	Management	Report Server	The Report Server failed to encrypt values for a setting in a configuration file.

Reporting Services Trace Logs

The Reporting Services trace logs contain information that is also recorded in other trace logs, but with added information that you will not find anywhere else. These logs are useful for troubleshooting custom applications that are integrated with Reporting Services or if you need more detail about an event that was written to another log.

Trace logs are limited to 32MB in size and they are deleted every 14 days by default. New trace logs are created every day, starting with the first event that occurs after midnight, local time, in the `C:\Program Files\Microsoft SQL Server\`*`Instance`*`\Reporting Services\LogFiles` directory. The name for each log file is derived from the local time on the computer. There are four trace logs to work with:

- `ReportServerService_`*`timestamp`*`.log` contains data on the Report Server Windows service worker threads.
- `ReportServerService_main_`*`timestamp`*`.log` contains data on the Report Server Windows service management threads.
- `ReportServerWebApp_`*`timestamp`*`.log` houses information about Report Manager.
- `ReportServer_`*`timestamp`*`.log` holds data regarding the Report Server Web service.

The trace logs are just text files that you can open with any text editor, such as Notepad. Using these logs, you can find out whether a report delivery occurred as expected, who received a report, how many delivery attempts were made, and so on. You can configure how much data is stored in the trace logs by configuring the `DefaultTraceSwitch` and `RSTrace` settings in the appropriate configuration file.

- The `ReportServerService_`*`timestamp`*`.log` and `ReportServerService_main_`*`timestamp`*`.log` files are configured in the `ReportingServicesService.config` file.
- The `ReportServerWebApp_`*`timestamp`*`.log` file is configured using the `Web.config` file for Report Manager.
- The `ReportServer_`*`timestamp`*`.log` is configured in the `Web.config` file for the Report Server Web service.

The `DefaultTraceSwitch` determines what kind of information is recorded in the log file. There are five possible settings:

- 0 disables tracing.
- 1 traces exceptions and restarts.
- 2 traces exceptions, restarts, and warnings.
- 3 traces exceptions, restarts, warnings, and status messages (this is the default setting).
- 4 sets verbose tracing.

The `RSTrace` setting configures the mechanical aspects of the trace log, such as where it is stored and how big it can get. The settings are listed in Table 10.4.

The configuration file should look something like this:

```
<system.diagnostics>
 <switches>
     <add name="DefaultTraceSwitch" value="3" />
   </switches>
 </system.diagnostics>
```

```
  <RStrace>
     <add name="FileName"
value="ReportServerService_" />
     <add name="FileSizeLimitMb" value="32" />
     <add name="KeepFilesForDays" value="14" />
     <add name="Prefix" value="tid, time" />
     <add name="TraceListeners" value="debugwindow,
file" />
     <add name="TraceFileMode" value="unique" />
     <add name="Components" value="all" />
  </RStrace>
```

TABLE 10.4: *RSTrace* Settings

SETTING	DESCRIPTION
FileName	Defines the portion of the filename before the last underscore.
FileSizeLimitMB	Defines the maximum size for the trace log.
KeepFilesForDays	Specifies how many days a trace log can remain on disk before it is deleted.
Prefix	Defines the portion of the filename after the last underscore.
TraceListeners	Specifies the target for the trace log output. Valid values are DebugWindow, File, and StdOut. DebugWindow and File are the defaults.
TraceFileMode	Do not modify this value; it is for internal use. It specifies whether the trace log should contain data for a 24-hour period.
Components	Defines which components are traced and at what level. You should not modify this value in the ReportServerService.config file or the Web.config for Report Manager. In the Web.config for Report Server, you can change this using a comma-separated list of components and their trace levels. The trace levels are 0–4 (the same as DefaultTraceSwitch). The components are: All is used to trace all processes that are not grouped into specific categories. RunningJobs traces an in-progress report or subscription. SemanticQueryEngine traces ad hoc queries performed against models. SemanticModelGenerator traces model generation.

REPORT SERVER EXECUTION LOGS

The Report Server Execution log contains information about the reports that execute on a server, or multiple servers in a scaled-out deployment. You can find information about how often a specific report is executed, what the most popular file formats are, and how long the

report takes to process in milliseconds. Specifically, this information is captured in Report Server Execution logs:

- The name of the Report Server instance that handled the request
- Report identifier
- User identifier
- Request type (user or system)
- Rendering format
- Parameter values used for a report execution
- Start and stop times that indicate the duration of a report process
- Number of milliseconds spent retrieving the data, processing the report, and rendering the report
- Source of the report execution (1 = Live, 2 = Cache, 3 = Snapshot, 4 = History)
- Status (either rsSuccess or an error code; if multiple errors occur, only the first error is recorded)
- Size of rendered reports in bytes
- Number of rows returned from queries

You can enable or disable report execution logging in Report Manager on the Site Settings page. You can also specify how long you want to keep the log file entries, which is 60 days by default. Entries that exceed the maximum are deleted at 2:00 AM every day.

Unlike other logs, the report execution log is not stored in a file on disk; it is stored in a number of tables that are scattered throughout the system, which makes the log difficult to read. To help you get useful information from the log in a format you can work with, Microsoft has provided a SQL Server Integration Services package that extracts all the report execution log data and places it into a table structure. The files that you need to create the database and extract the data are `RSExecutionLog_Update.dtsx`, `RSExecutionLog_Update.dtsConfig`, and `Createtables.sql`.

Reporting Services also comes with some sample reports that display the data collected from the report execution log. These are `Execution Status Codes.rdl`, `Report Summary.rdl`, and `Execution Summary.rdl`. All of the sample reports and SQL Server Integration Services (SSIS) packages are installed with the SQL Server samples.

Tips for Common Problems

There are a lot of components in Reporting Services that can potentially cause problems, and some of those problems are fairly common. So, in this section we will discuss some of the more common problems and how to troubleshoot them.

HTTP Problems

HTTP errors are generated by Internet Information Services (IIS), not Reporting Services. There are some common error codes that you might see.

HTTP code 500 is a generic error that occurs when a server is not configured properly. If you encounter error 500, make sure that your Report Server web applications are configured to use `ASP.NET` 2.0 by checking the `ASP.NET` tab of the website properties page.

HTTP code 401 indicates a security configuration problem on the IIS website. There are several reasons this can happen; the most common involves service pack issues.

If you are running Reporting Services on a Windows XP machine with Service Pack 2 installed or on a Windows Server 2003 machine with Service Pack 1 and you try to access the Report Server locally using a fully qualified domain name or a custom header, then you may see error 401. This is because of a new loopback check security feature that is included in the service packs. There are two workarounds for this:

This first method is to disable the loopback altogether:

1. Click Start ➢ Run, type **regedit**, and then click OK.
2. In Registry Editor, locate and then click the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Control \Lsa
```

3. Right-click Lsa, point to New, and then click DWORD Value.
4. Type **DisableLoopbackCheck**, and then press Enter.
5. Right-click DisableLoopbackCheck, and click Modify.
6. In the Value Data box, type **1**, and then click OK.
7. Quit Registry Editor, and then restart your computer.

The second method involves adding the hostnames that you want mapped to the loopback adapter so they can connect to websites on the local machine:

1. Click Start ➢ Run, type **regedit**, and then click OK.
2. In Registry Editor, locate and then click the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\
Control\Lsa\MSV1_0
```

3. Right-click MSV1_0, point to New, and then click Multi-String Value.
4. Type **BackConnectionHostNames**, and then press Enter.
5. Right-click BackConnectionHostNames, and click Modify.
6. In the Value Data box, type the hostname(s) for the sites that are on the local computer, and then click OK.
7. Quit Registry Editor, and then restart the IISAdmin service.

Report Problems

One of the problems you may run into with reports occurs when you export a report to a web archive. If you export a report to a web archive file and open it with Internet Explorer (IE), it looks fine the first time you open it. Later, you may decide to change some parameters for the report and export it again. When you open the report again, it hasn't changed to reflect the new parameter values. This is because IE is opening the report from its local cache. To work around this issue, clear your local cache and open the report again.

Another common problem occurs with reports that have toggled items. You may generate a report that contains toggled items, but when you click the + icon to expand the item,

nothing happens. This is probably because the name of the Report Server where you generated the report contains an underscore (_) character. Reporting Services does not support underscores in the server name, and there is no workaround for this.

Report Processing Problems

When you request a report in Report Manager and Reporting Services has trouble generating the report for some reason, you will see an error message on the screen instead of the report you were expecting. If you are logged on to Report Manager using local administrator credentials, you can right-click on the page and select the View Source option to view the call stack, which contains more information about the error.

You can also use the Run As command to execute a report using a different set of security credentials. If you use this command on a report that includes charts, it may not run correctly. This is because the Report Server caches charts as temporary Internet files and the credentials you are using to run the report may not have access to read those temporary files. To fix the issue, grant the account you are using access to read the temporary Internet files.

Note

For those of you who are not familiar with Windows administration, you can access the Run As command by right-clicking a file and selecting Run As, or by using the `runas` command from the command prompt.

Connection Problems

The most common connection error that comes up is ''Cannot create a connection to datasource *datasourcename*.'' This is always accompanied by a secondary message that describes the underlying cause of the problem.

One of the messages is ''Login failed for user 'NT AUTHORITY\ANONYMOUS LOGON'.'' This occurs when you do not have Kerberos security enabled on your network and you are passing credentials across more than one computer. For example, suppose you are connected to a Report Server named CorpReports, which connects to a SQL Server named CorpSQL. The report you are trying to run connects to a SQL Server named AcctgSQL. In this case, you would be trying to pass credentials from CorpReports to CorpSQL, which works fine. The next step, passing credentials from CorpSQL to AcctgSQL, fails and Reporting Services displays an error (see Figure 10.1).

Another message you may see is ''The Report Server cannot open a connection to the Report Server database. A connection to the database is required for all requests and processing (rsReportServerDatabaseUnavailable).'' That is a mouthful, and it may simply mean that the database server that houses the Report Server database is not configured to accept remote connections. Follow these steps to configure the SQL Server to allow remote connections:

1. Click Start ➢ Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ SQL Server Surface Area Configuration Tool.
2. Click Surface Area Configuration For Services And Connections.
3. Open the folder for the SQL Server instance that hosts the Report Server databases.
4. Click Remote Connections.
5. Click Local And Remote Connections.

6. Click Using Both TCP/IP And Named Pipes.
7. Click OK, and then restart the service.

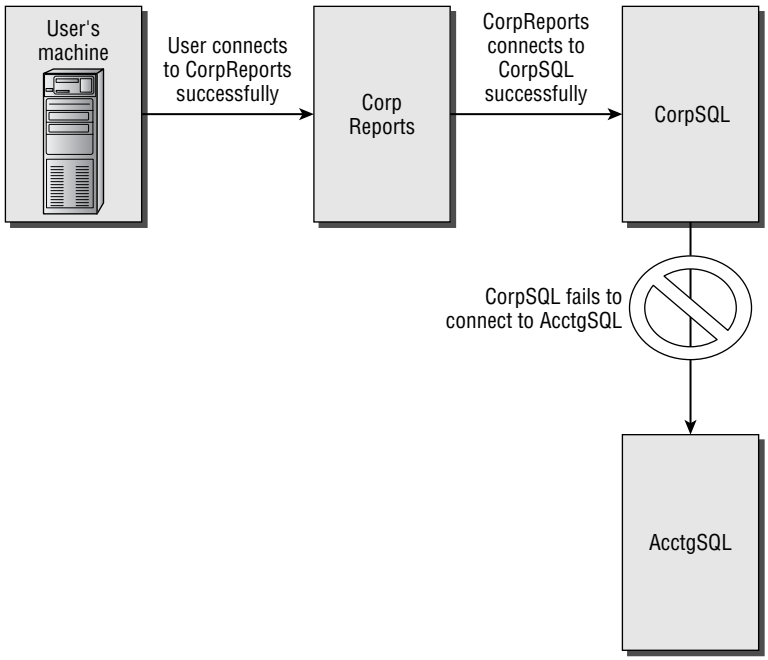


FIGURE 10.1
Report execution will fail if you are passing credentials across more than one server without Kerberos security enabled.

If the error is accompanied by this additional text:

```
An error has occurred while establishing a connection to
the server. When connecting to SQL Server 2005, this
failure may be caused by the fact that under the default
settings SQL Server does not permit remote connections.
(provider: SQL Server Network Interfaces, error: 26 - Error
Locating Server/Instance Specified).
```

then the password has expired on the account used to run the Database Engine instance and you need to reset the password.

Report Manager Problems

You may be greeted with a blank page when you try to open Report Manager. If this is the case, you probably have anonymous access enabled for the Report Manager website in IIS. Enabling anonymous access is tantamount to shutting off role-based security, so Report Manager will not work. You need to disable anonymous access to fix the issue.

If it is only the New Folder, New Data Source, Upload File, and Help pages that do not open, you must modify your browser settings in Internet Explorer. Here is what you need to do in IE 7:

1. Click Tools ➢ Internet Options.
2. One the Security tab, click the Trusted Sites zone.

3. Click the Sites button.
4. Type the name of your server in the text box. It should be in URL format: `http://`*`ReportServerName`*.
5. Click Add.
6. Click Close.
7. Click OK.

Subscription and Delivery Problems

Subscriptions can fail under a number of conditions, including:

- The schedule used to trigger the report or to schedule a snapshot update may have expired.
- The SQL Server Agent or Report Server is not running.
- The report may be too large to deliver via e-mail. You can test for this scenario by saving the report to a file and trying to e-mail it manually.
- The computer that houses the network share used for file-share delivery is not running.
- The delivery extension specified in the subscription has been uninstalled or disabled.
- The credential settings have been changed from stored to integrated or prompted values.
- The parameter name or data type was changed in the report definition, and the report was republished.

Another error you may run into when using the e-mail delivery provider is ''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.'' This error means that your e-mail server is not configured to allow e-mail submission from anonymous users. Ask your e-mail administrator to configure the e-mail system to allow anonymous access to fix this issue.

Initialization and Encryption Problems

Initialization errors occur when you first install Reporting Services. You should expect to get an initialization error when you try to join a new Report Server to an existing farm. If you get an initialization error on a single server installation, make sure that the account used to run the Report Server Windows service has full control on the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Reporting Services` key in the registry.

If you change the profile information of the Report Server instance, then you may get this error: ''The Report Server Web service is not running as the Web service account specified in rsreportserver.config.'' Follow these steps to try to resolve this issue:

1. Run the Reporting Services Configuration tool to update the Report Server database connection information.
2. Restore the encryption keys to update the key information.

Configuration Problems

One of the configuration problems that you may get pops up after installing SQL Server 2005 Service Pack 2. The Reporting Services Configuration tool that comes with Service Pack 2 (SP2) uses a new version of the Windows Management Instrumentation (WMI) interface that is not

backward compatible with previous versions. So, if you try to configure a pre-SP2 server with the new Reporting Services Configuration tool, you will receive an error that states, ''The WMI Provider has returned an unspecified status at ReportServicesConfigUI.WMIProvider. ConfigurationItem.CreateConfigurationItem.''

Another common issue is that `ASP.NET` may time out after two minutes of report processing when the server is installed in SharePoint integrated mode. To address this issue, configure the `executionTimeout` setting in the SharePoint `Web.config` file by following these steps:

1. Go to *x*`:\inetpub\wwwroot\wss\VirtualDirectories\`*directory*. The directory name is based on the port number used to access the SharePoint site. If the SharePoint site uses port 80, the path will be `\inetpub\wwwroot\wss\VirtualDirectories\80`.
2. Open `Web.config` and search for `httpRuntime`.
3. Add the `executionTimeout` value. To increase the value to 9000 (which is the default Reporting Services setting), use the following code:

```
<httpRuntime maxRequestLength="51200" executionTimeout="9000" />
```

The Bottom Line

Manage the Windows and web services. Reporting Services is made up of a Windows service and a web service. The Windows service performs initialization, encryption, database maintenance, and scheduling and delivery functions. The web service processes ad hoc reports and provides developers with a programming interface so they can integrate Reporting Services into their applications.

> **Master It** You work for a large organization that uses Reporting Services heavily. After taking inventory of your hardware, you are convinced that you need to distribute Reporting Services functionality across multiple servers. How can you do this?

Back up and restore the Report Server database. There are a number of valid reasons to back up the databases that support Reporting Services, such as theft, natural disaster, and hardware failure. You should back up two databases: the ReportServer database and the ReportServerTempdb database. The ReportServerTempdb database does not hold persistent data, so you only need to back up the structure. You need to back up the data and the structure of the Report Server database.

> **Master It** You have just installed Reporting Services on a new server and you decide to perform a backup to protect your hard work. You try to perform a transaction log backup of your ReportServer database, but it fails. Why does it fail, and what can you do to fix the problem?

Manage encryption keys. When the server is initialized, it acquires a symmetric key from Windows Cryptographic Services that it can use to encrypt and decrypt data. The information used to connect to the Report Server database is encrypted using the machine-level key and stored in the `RSReportServer.config` file. Stored credentials used by data sources are encrypted using the symmetric key. The unattended user account information is encrypted and stored in the `RSReportServer.config` file. Finally, the symmetric key itself is encrypted.

Master It After installing and initializing your Report Server, you realize that you need to back up the encryption keys to make sure that they are not lost. You decide to back them up to a floppy disk that you can keep off-site for added protection. How do you do this?

Manage report processing and content. One of the primary functions of Reporting Services is to store content such as reports, models, folders, resources, and shared data sources. As an administrator, you need to manage this content.

Master It One of your users requested a report through Report Manager about 10 minutes ago. The report usually only takes about 3 minutes to execute. After investigation you find that the process is stalled. What can you do to fix the problem?

Troubleshoot common problems. A number of problems can arise in Reporting Services, including issues related to HTTP, reports, processing, connections, Report Manager, subscription and delivery, initialization and encryption, and configuration.

Master It You have just installed Reporting Services on the same server as your database engine and everything seems to be working fine when you are connected to the machine locally. When you attempt to access the server over the network you get an error that reads, ''Cannot create a connection to datasource *datasourcename*.'' It is accompanied by another message that reads, ''The Report Server cannot open a connection to the Report Server database. A connection to the database is required for all requests and processing (rsReportServerDatabaseUnavailable).'' What should you do to fix the issue?

Part 3

Advanced Topics

We get into some of the truly advanced Reporting Services topics — not for the faint of heart, we warn you — in this part.

Reporting Services provides a robust application programming interface (API) that developers can use to integrate Reporting Services with their own custom applications. In the first three chapters of Part 3, I'll show you how to put that API to use in different circumstances.

This part is not just for developers, though; there is plenty of advanced material for system administrators as well. I'll show you how to secure your Report Server, create data sources and data sets for third-party data stores, and tune the performance of your Report Server.

There are six chapters in this section:

- **Chapter 11: Managing Reporting Services Using Program Code**
- **Chapter 12: Extending Reporting Services**
- **Chapter 13: Report Definition Language**
- **Chapter 14: Report Server Security**
- **Chapter 15: Data Sources and Data Sets**
- **Chapter 16: Performance Tuning**

Chapter 11

Managing Reports Using Program Code

In addition to accessing the Report Server through your web browser and deploying new reports from within the Intelligence Development Studio, you can access the Reporting Services functionality through the appropriate APIs, which you can use to either integrate Reporting Services reports in custom Windows and web applications or to extend the Report Server's functionality. In this chapter you'll find an overview of the Reporting Services APIs, learn how to access a Report Server (and its contents) from within custom Windows and web applications, and see how to manipulate the Report Server's objects from within custom applications.

In short, there are two basic operations you can perform through the Reporting Services API: manage the contents of the Report Server (from moving items around to creating new reports in code) and request the rendering of existing reports. After reading this chapter you'll know how to incorporate the functionality of Reporting Services in custom applications, as well as how to develop custom tools for managing the Report Server. Note that there's nothing you can do from within a custom application that you can't do through Report Manager, but you'll be able to develop interactive applications to manage the Report Server's content, as well as request reports and display them on Windows forms.

In this chapter you will learn to

- Access reports with a URL
- Pass commands to the Report Server through a URL
- Manage reports with the `rs` command-line utility
- Use the Report Server web service
- Integrate reports into Windows and web applications

Accessing Reports from within Your Code

There are basically two techniques for connecting to a Report Server programmatically:

The Reporting Services web service The Report Server exposes its functionality through the Reporting Services web service. After referencing it from within your application, you can call the methods of the service to manage the objects on the server (create new folders, roles, reports; organize these objects; and so on), as well as execute reports on the server and retrieve the results in your application. For example, you can provide a rich Windows application to collect the parameter values for a report, pass them to the Report Server, execute the report, and finally display the result in a WebBrowser control in your custom application. You can even retrieve the report as an image and store it or transmit it to a remote client that has no direct access to your Report Server.

URL-based commands You can pass URL-based commands to the Report Server to execute specific reports in a browser window. In addition to making URL requests to a Report Server from within a browser, you can use the same requests in conjunction with the WebBrowser control to include reports in Windows and web forms.

Accessing Reports with a URL

Administrators should be able to launch reports without using Report Manager and perform basic maintenance tasks on the Report Server with command-line tools. Reporting Services exposes its functionality through the browser. You can enter the URL of a report in the address bar to render a report in your browser. You can also specify parameter values and many formatting options. If you store these URLs as hyperlinks in shortcuts on the users' desktops, they can open reports like any other web page (as opposed to locating the desired report through the Report Server's interface).

Viewing Reports

You can access your Report Server from your web browser with a URL that looks like the following:

```
http://<Your Server>/ReportServer/<Report Name>
```

where the names in angle brackets, <Your Server> and <Report Name>, must be replaced by your server name and the report you want to view, respectively. The <Report Name> placeholder must be set to the path to the report relative to the home directory. It could be either the report name or one or more folder names followed by the report name, as in

```
http://localhost/ReportServer/AWCustomers/AWCustomerAddresses
```

If you omit the report name, the Report Server will return a page with the names of the items in its root folder (folders, reports, and data sources). Of course, you'll only see the items you've been authorized to view. If you examine the source code of a page that displays report names, you'll see that the hyperlink for any report looks like the following:

```
<a href="/Reports/Pages/Report.aspx?➢
   ItemPath=/AWCustomers/AWCustomersPerCountry">
```

The parameter *ItemPath* specifies the path to the selected report. If you paste this URL into the browser's address bar, the slash characters will be replaced by the string `%2f`, which is the hexadecimal representation of the slash character. Other special symbols, such as spaces, commas, and so on, are replaced by hex values (URL-encoded values).

The URL-based method shouldn't come as a surprise to you, as you already know how to request individual reports from the Report Server: just open a report and copy its URL from the browser's address box. You can use this URL as a hyperlink in a web page, or pass it to the `Navigate` method of the WebBrowser control to display the report in a Windows application. To see this technique in action, direct your browser to a report on the Report Server and when the report appears in your browser, copy the URL. Here's what a report's URL looks like:

```
http://ReportServer/Pages/ReportViewer.aspx?/AWCustomers/➢
   AWCustomersPerCountry&rs:Command=Render
```

Using Reporting Services with Multiple Instances of SQL Server 2005

If you're using Reporting Services on an instance of SQL Server other than the default one, you should append the name of the instance of SQL to the `ReportServer` keyword with a dollar sign as a separator (the instance of SQL Server here is `RSServer` and `PowerKit` is the name of my server):

```
PowerKit/ReportServer$RSServer
```

Because we've used the `RSServer` instance of SQL Server 2005 on the `PowerKit` server, you'll see the string `PowerKit/ReportServer$RSServer` in many of the figures. If you're using the default instance of SQL Server 2005 on your computer, this is equivalent to `localhost/ReportServer`.

This URL requests the AWCustomerPerCountry report in the `AWCustomers` folder under the Report Server's root folder. The parameter `rs:Command=Render` of the URL is actually a command that instructs the server to render the report as a web page. You can send additional commands to the server, as you'll see later in this chapter, but let's see first how we can reuse this URL.

Start Visual Studio, create a new Windows application, and place an instance of the WebBrowser control on the form. Don't forget to set the control's `Dock` property to `Fill`, so that the control covers the available client area. In the form's `Load` event handler enter the following statement:

```
Private Sub Form1_Load(ByVal sender As System.Object,➢
        ByVal e As System.EventArgs) Handles MyBase.Load
    WebBrowser1.Navigate(➢
           "http://Server/ReportServer/Pages/➢
            ReportViewer.aspx?/AWCustomers/➢
            AWCustomersPerCountry&rs:Command=Render")
End Sub
```

You must replace the string *`Server`* with the name of the Report Server you want to access, most likely the string `localhost` and the sample report's path with an actual report on your server.

If you aren't using Visual Studio, you can do something similar with a web page. Create a new web page and insert a hyperlink to the same URL. Start Notepad and enter the following HTML code. Then save the document as `URLAccess.html`.

```
<html>
<h1>URL Access to Report Server</h1>
You can view <a
href="http://Server/ReportServer/Pages/ReportViewer.aspx?/➢
   AWCustomers/AWCustomersPerCountry&rs:Command=Render">a➢
   sample report</a> on the local Report Server.
</html>
```

As you'll see shortly, there are quite a few ways to customize the behavior of the report from within its URL. You can't actually change the appearance of the report, but you can control the rendering of the report (you can request an XML file or an image of the report), show or hide certain sections, pass report parameters without user intervention, and so on.

If you click on a folder that contains other folders and/or reports, the browser will submit a URL to the Report Server to retrieve the list of items. This URL will look like the following:

```
http://localhost/ReportServer/Manager?/AWCustomers&
rs:Command=ListChildren
```

The parameter `rs:Command=ListChildren` is called a *command parameter*. It's actually a command that instructs Reporting Services to return the list of items under the current folder. Reporting Services recognizes two basic command prefixes in the URL: the `rc` and `rs` parameter prefixes, which can be followed by a command. To submit multiple commands, you just repeat the appropriate prefix. The `rs` prefix is used to pass commands to the Report Server, while the `rc` parameter is used to pass commands to the rendering engine. One of the `rs` commands is the `Render` command, which displays the report. You can control the appearance of the report by passing a command to the renderer. The following URL displays a report without the area that prompts the user for the report's parameters:

```
http://localhost/Reports/AWCustomers/CustomerOrders?
rs:Command=Render&rc:Parameters=False
```

The question mark separates the destination URL from its parameters, while the ampersand separates parameters. With the `rs` parameter we pass a command to the Report Server to invoke the report (the `Render` command) and with the `rc` parameter we pass a command to the renderer of the Report Server to hide the parameter prompt section at the top of the report. You may be wondering why the `Render` command is passed with the `rs` prefix; it's because the command is executed by the Report Server. The rendering engine handles the details of the rendering of the report, but doesn't actually produce it.

Using the *rs:Command* Parameter

The `rs:Command` parameter recognizes the following commands for the Report Server:

`ListChildren` Lists the items in the current directory. The `ListChildren` command returns all the items, whether they're folders, reports, or resources. The list returned by the Report Server is an HTML page.

`Render` Launches and displays the report.

`GetDataSourceContents` Returns the contents of the selected data source. It doesn't return the actual data, but just the connection string.

`GetResourceContents` Returns the contents of the selected resource (like a JPEG or a text file).

`Format` The `Format` command requests that the report be rendered in a specific format. This option is usually omitted and the renderer emits the proper HTML file based on the browser-detection capabilities of the browser that made the request. The `Format` command recognizes the following values (they're the same ones you see in the `Export` command of the browser when you request a report):

- `XML`: Returns the report data in XML format. The file returned by the server doesn't contain any formatting information, but just data.
- `CSV`: Returns the report data as a list of comma-separated values.

- `IMAGE`: Renders the report as an image. Long reports are rendered in multiple pages and you can request individual pages.
- `PDF`: Renders the report in PDF format.
- `HTML4.0`: Generates an HTML 4.0 file with the report.
- `HTML3.2`: Generates an HTML 3.2 file with the report.
- `MHTML`: Returns the report data as an MHTML file.
- `EXCEL`: Returns the report data in Excel format.

The `rs:Command` and `rs:Format` are the two basic command parameters you can pass to the Report Server. There are a few more `rs` command parameters, which are usually omitted, but here they are:

`ClearSession` Set this parameter to `True` to generate a new report with up-to-date data, instead of getting it from the history snapshot.

`ParameterLanguage` Specify the language used in passing URL parameter values to the Report Server. This setting is a string like `en-us`.

`Snapshot` The report is not created from live data, but it's fetched from the history snapshot.

The *rs* Utility

A handy tool for managing the Report Server is the `rs` utility; `rs.exe` is a command-line tool that allows you to perform many management operations against the Report Server using scripts. These scripts are Visual Basic `.NET` files with the `.rss` extension. The script file is a Visual Basic module that must contain a `Main()` subroutine (this is the module's startup object), and it may contain other subroutines and functions. The `rs` utility connects automatically to the Report Server, creates the proxy class, and exposes the `rs` variable that represents the proxy. This is a module-level variable, which you can use from within the script to call the methods of the `ReportingService2005` class.

The options available through the `rs` command can also be executed against the Report Server with the `rs.exe` command-line utility. The script files can be written in any `.NET` language, and they allow you to access the functionality of the Report Server through the `rs` variable (you don't have to declare or instantiate it). However, you must supply the proper credentials to access the Report Server, and the simplest method to do so is to set the `Credentials` property. Here's a simple script that lists the items in the Report Server's root folder:

```
Public Sub Main()
rs.Credentials = System.Net.CredentialCache.DefaultCredentials
Dim items() As CatalogItem
items = rs.ListChildren("/", False)
Dim item As CatalogItem
For Each item In Items
    Console.WriteLine(item.Name)
Next
End Sub
```

As you can see, calling the Report Server's method through the `rs` variable is more flexible than specifying parameters in the URL. The `ListChildren` method, for example, accepts two arguments that let you specify the parent folder and whether or not the method will return the

names of nested objects (in the sample code, the second argument is set to `False` to prevent the recursive display of the items). To test the previous script, store the statements in a text file with the `.rss` extension and then open a command window and execute the following statement:

```
rs -i sample.rss -s http://localhost/ReportServer
```

The `-i` switch specifies the input file (the RSS script file), and the `-s` switch specifies the server against which the script will be executed. The `rs` utility recognizes a few more switches, like the `-u` and `-p` switches that specify a username and password for the server, the `-l` switch that specifies the timeout in seconds and the `-v` switch, which must be followed by key/value pair in the following format:

```
-v variable_name=variable_value
```

Use this switch to pass parameters to the script. The parameters in the script need not be declared; you just call them in your code. However, they must be enclosed in parentheses. To pass the `var1` variable to a script, call the `rs` utility as follows:

```
rs -i sample.rss -s http://localhost/ReportServer -v var1 = 2007
```

In the script's code you can retrieve the value of this variable as long as you enclose it in parentheses:

```
Console.WriteLine (var1)
```

If you have multiple script parameters, prefix each one with the `-v` switch.

Let's revise the previous script a little so that it will return some additional information about the items under the specified folder. The `ListChildren` method returns an array of `CatalogItem` objects, and these objects can be data sources, folders, linked reports, reports, and resources. The information about the item's type is stored in the `CatalogItem` object's `Type` property. The following code segment displays the names of all objects in the root folder of the Report Server:

```
Public Sub Main()
Dim RS As New ReportService.ReportingService2005
RS.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim items() As ReportService.CatalogItem
items = RS.ListChildren("/", True)
Dim item As ReportService.CatalogItem
For Each item In items
    Select Case item.Type
        Case ReportService.ItemTypeEnum.DataSource
            Debug.WriteLine("DATASOURCE: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
        Case ReportService.ItemTypeEnum.Folder
            Debug.WriteLine("FOLDER: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
```

```
        Case ReportService.ItemTypeEnum.LinkedReport
            Debug.WriteLine("LINKED REPORTS: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
        Case ReportService.ItemTypeEnum.Report
            Debug.WriteLine("REPORT: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
        Case ReportService.ItemTypeEnum.Model
            Debug.WriteLine("REPORT: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
        Case ReportService.ItemTypeEnum.Resource
            Debug.WriteLine("RESOURCE: " & item.Name)
            Debug.WriteLine("            " & item.CreationDate)
        Case Else
            Debug.WriteLine("UKNOWN ITEM: " & item.Name)
        End Select
    Next item
End Sub
```

The (partial) output generated by this code segment looks like the following:

```
FOLDER: AWCustomers
            1/28/2007 5:40:25 pm
REPORT: AWCustomersPerCountry
            1/28/2007 5:40:28 pm
FOLDER: Data Sources
            1/28/2007 5:40:26 pm
DATASOURCE: AWSource
            1/28/2007 5:40:27 pm
```

Unlike standard `.NET` modules, RSS scripts can't contain `Imports` statements to reference `.NET` namespaces. However, the scripting environment exposes on its own several useful `.NET` namespaces such as `System.Xml` and `System.IO`.

Now, is there something really practical you can do with the `rs` utility? Given that scripts are inflexible in terms of user interaction, you can create scripts to perform management operations. Writing a script that copies a report from one server (the development server) to another one (the production server) is actually quite trivial, and you'll see the code for this script later in the chapter. You can also copy or move reports from one folder to another with simple scripts. However, you can't pass the `Render` command to invoke a report with the `rs` utility.

Using the *rc* Parameters

The `rc` prefix recognizes many more options. The `rc` command parameters are passed to the renderer of the Report Server and they affect the appearance of the report (and no, there's no command-line tool equivalent to the `rs` utility for scripting `rc` commands). Three command parameters control the visible areas of the report:

`Toolbar` Set it to a `True/False` value to indicate whether the toolbar of the report will be displayed (when the report is rendered in HTML format).

Parameters Set it to a `True/False` value to indicate whether the parameters section of the report will be displayed. You can also set this parameter to `Collapsed`, which means the parameters section will be hidden but that users will see a button they can click to expand this section if they want.

DocMap Set it to a `True/False` value to indicate whether the document map of the report will be displayed.

Other commands determine the size of the browser window and the navigation within the report:

Zoom A numeric value that determines the report's zoom factor. You can also set it to one of the constants `Page Width` or `Whole Page`.

Section A numeric value that determines the section or page of the document to navigate to.

LinkTarget This parameter controls the window when the user navigates to subreports of the main report, and its value can be one of the following constants: `_top`, `_self`, `_search`, `_parent`, or `_blank`. It can also be set to the name of a window that you have already created.

Finally, there's another set of command parameters that allow you to search for a specific string within a report (the search operation is also available through the report's toolbar in the browser):

FindString The string to search for

StartFind, EndFind The starting and ending page for the search

FallbackPage The page that will be displayed should the search fail

Passing Report Parameters in a URL

When a report with parameters is invoked, the web page contains the proper controls where users can specify the values of the parameters and then render the report. It's also possible to pass parameter values in the URL and hide the report's interface so that users will see the report right away. You can use this technique to call reports from within custom applications or other web pages. If you have designed a page that collects information from the user, you can create a hyperlink that, in addition to the report's name, contains the values of the necessary parameters. If the user has selected an employee to view that employee's data, the hyperlink to the EmployeeSales report on this page need not prompt the user for the employee name or ID, because the information is already available. Here's how you'd pass the employee ID and a default period for the report:

```
http://localhost/ReportServer?/AWReports/EmployeeSales&
rs:Command=Render&EmpID=31 & Quarter=2&Year=2007
```

Parameters do not take a prefix; everything that's not prefixed by the `rs` or `rc` qualifier is a parameter name and should be followed by the equal sign and a value. You must pass values for all parameters, with the possible exception of the parameters with default values.

To hide the report's interface that prompts for parameter values (the section with the various controls at the top of the report), you must tweak the report's properties. Go to the report's page by using Report Manager and switch to the Properties tab. Select the Parameters section and check the Hide box, as shown in Figure 11.1. Then click Apply and the report will not prompt users for parameter values. If you don't want to permanently change the report's behavior, include the

`rc:Parameters = False` parameter in the URL. Even better, use the value `Collapsed` for this parameter, so that users can expand this section if needed.

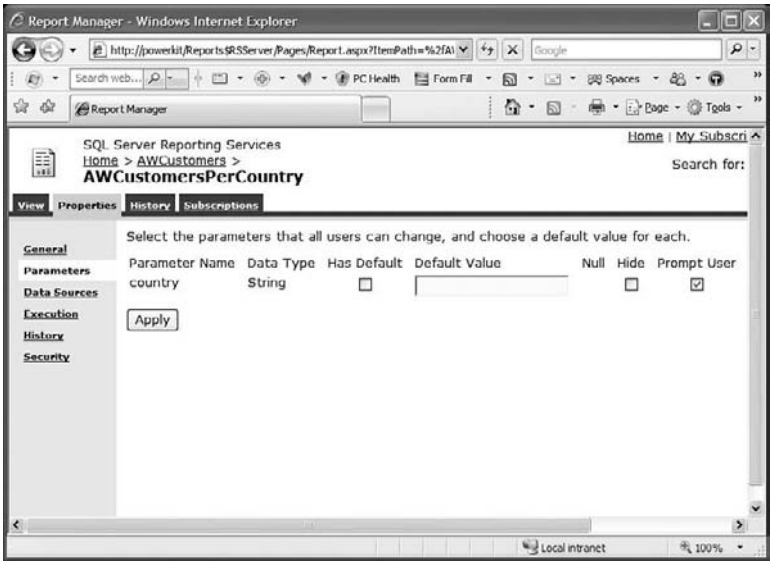


FIGURE 11.1
The basic settings of the report's parameters

The Report Server Web Service

The Report Server exposes its functionality through the Report Server web service. This web service is made up of two classes: one for executing reports and one for managing reports. Note that the web services reside on the same server as the Report Server; if you execute the same application against a different server, you must add a reference to the web service exposed by the new server. You can use this web service to develop Windows applications that access the Report Server and manage or execute reports.

If you're not familiar with web services, they're components that run on the same server as the Report Server and they expose methods, similar to `.NET` classes. An application running on a workstation contacts the web service and requests the execution of a specific method. You already know that you can pass the `Render` command to the Report Server as a URL parameter. The same command is also available through the web service. Assuming that you have a reference to the web service, you can use it to access the methods exposed by the service. Calling a web service method is identical to calling any other object's method. The reference to the web service is nothing more than a variable, say the `RS` variable. Once you have declared the `RS` variable, you can call the `Render` method with the following statement (followed by the proper arguments, as you will see shortly):

```
RS.Render
```

The web service runs on a remote computer, which may be located halfway across the globe. The variable that represents the web service makes it seem as if the web service were a local component. You will see in the following section how to set up a variable that represents a web

service (a *proxy*). Once the proxy is in place, you can program against Reporting Services as if it were a local component running at the client. The transmission of the parameters to the method you're calling, the interception of the results, and the very fact that the code of the method is being executed remotely are all transparent to your application.

But why access the Report Server through a web service in the first place? The Report Server runs on a server machine and services requests from many workstations. You need a way to execute programs remotely: send a request from a client to the server, have the server execute some code, and return the result to the client. The result is the report in various formats, or simply the data that make up the report. And why can't we use the browser to view reports? Once the report has been rendered in the browser's window, you can export it in many formats with the Export command in the toolbar. The browser-based interface provides an identical interface for all reports. The web service allows you to include the full functionality of the Report Server into custom applications. You can write an elaborate interface using Visual Basic or any other language and include reporting capabilities with simple calls to the web service exposed by the Report Server. When the report arrives at the client, you can render it in a Windows form, grab the data and display it in any built-in or custom control, process the data in ways that are simply not possible with Reporting Services, and you can even store a report as a series of images (depending on the size of the report and the page size).

You can also use the web services exposed by Reporting Services to manage the reports on the server. You can write simple Windows applications that allow you to inspect the contents of the Report Server, move reports around, edit existing reports, and even create new reports from scratch all from within your code. This last capability isn't a trivial task, but you get the idea. It wouldn't be a bad idea to give the administrators of the Report Server a Windows application for performing the most common report-management tasks. The idea is that all of the functionality of the Report Server is exposed to developers, who can write highly interactive applications to manipulate the reports, from creating new reports to consuming existing ones.

We'll start with a detailed discussion of building Windows applications that harvest the functionality of the Report Server. By the way, the simplest method of presenting reports in Windows applications is to use the ReportViewer control. This control is a browser in disguise: it displays the report exactly as it would appear in Internet Explorer. You can still build a better navigational system for users to locate the desired reports and prompt them for the report parameter values with a rich client application, but you can't control the appearance of the report from within your code. In the remainder this chapter, you'll learn how to access the functionality of the Report Server directly and how to incorporate this functionality in your custom applications.

The Report Server web service provides two endpoints: one for rendering reports and another for managing report items. The management endpoint of the Report Server web service allows you to manage the objects on the Report Server from within custom applications, similar to managing them with scripts, as you saw earlier. The difference is that through the web service you can write interactive applications to do so. The management methods are contained in the class `ReportingService2005`. The execution endpoint of the same web service allows developers to programmatically control the processing and rendering of reports. The execution methods are contained in the class `ReportExecution2005`. You'll see examples of both endpoints shortly.

Instead of listing all available methods, which you can find in the Reporting Services documentation, let's start by building a VB application that uses the Report Server web service.

Accessing the Report Server from within .NET Applications

To use the web service from within a Windows application, you must first add a reference to the web service. Then declare a variable of the same type and use it in your code to access the methods

of the web service at the Report Server. This reference is a proxy — a class that communicates with the web service using SOAP (Simple Object Access Protocol) over HTTP — and it makes the web service's methods appear to the application as local methods. It maps the parameters you want to pass to the web service to XML elements and extracts the value returned by the web service (also as XML elements) into the appropriate types.

Of course, not anyone who knows the IP address of the Report Server can access the web service; the client application must be authenticated against the server with the credentials of a user that has been authorized to manage the server's objects or execute reports. Once the proxy is authenticated by the Report Server, the client application can call the methods of the web service.

Creating the Web Service Proxy

A client and a web service can communicate using SOAP messages, which encapsulate the input and output parameters as XML. A proxy class maps parameters to XML elements, which are then sent to the server over the network. In this way, the proxy class frees you from having to communicate with the web service at the SOAP level and allows you to invoke web service methods in any development environment that supports SOAP and web service proxies.

The simplest method to add a proxy class to a Windows project is to select the Add Web Reference command of the Project menu. When the Add Web Reference dialog box opens, click the link Web Services on the Local Machine (if you want to access the Report Server on the local machine) or the link Browse UDDI Servers on the Local Network to access the web services on another computer in the network. UDDI (Universal Description, Discovery, and Integration) servers catalog and organize web services. You can click this link to explore web services on other servers, as long as you know their names or IP addresses. You can also enter the complete path to the Report Server web service in the URL box at the top of the dialog box. Click the Web Services on the Local Machine link and you'll see the web services exposed by the local machine, as shown in Figure 11.2.

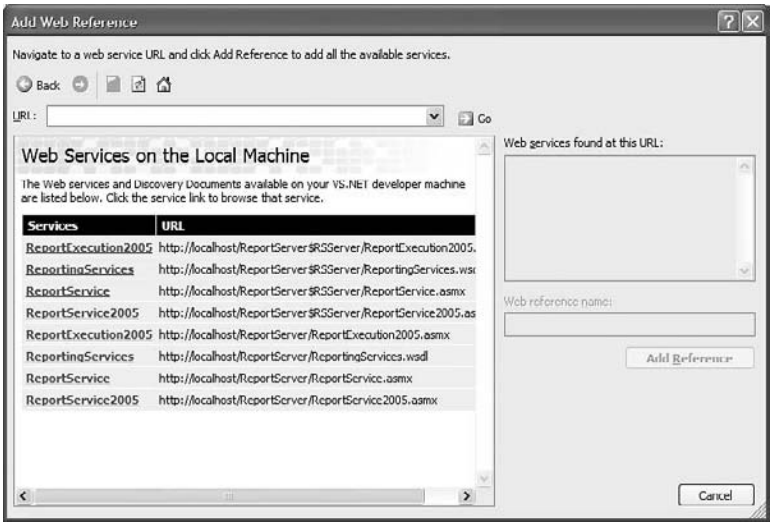
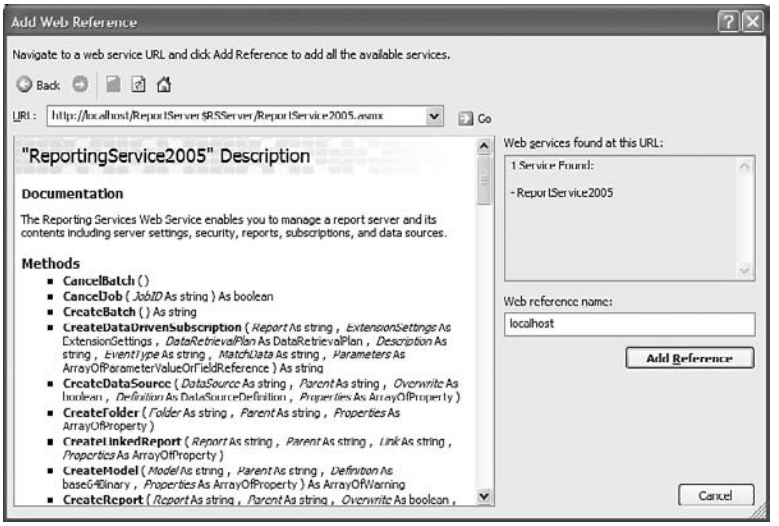


Figure 11.2
Locating the web services on the local machine

If the Report Server is not running on the local machine, set the URL box to the address of the Report Server. Locate the service ReportingService2005 and click its name. A few seconds later

you'll see the description of the ReportingService2005 web service. This description is a long list of the public methods exposed by the service, as shown in Figure 11.3. To add a reference to this service to the project, enter a name for the service in the Web Reference Name box and click the button Add Reference. A reference to the ReportingService2005 web service will be added to your project, and you can then create a proxy for accessing the Report Server.

FIGURE 11.3
Adding a reference to a web service



The ReportingService2005 web service exposes methods for manipulating the objects at the Report Server. To execute reports, add a reference to the ReportExecution2005 web service. Start a new project and add references to both web services, the ReportService, and ReportExecution references. Then place a button on the form and enter the following statements in the code window to create the necessary proxies to the web services. For all intents and purposes, the proxy is a typed object variable that exposes the same methods as the web service at the other end of the connection.

```
Dim RS As New ReportService.ReportingService2005
Dim RE As New ReportExecution.ReportExecution2005
```

The variables RS and RE should be declared outside any procedure, because in the sample project we want to be able to use them from within different modules. Add a button to the form and in its `Click` event handler enter the name of either variable followed by a period to see the list of the methods it exposes. These are the methods exposed by the appropriate web service, and the code is actually executed at the server. The client code calls these methods through the proxy, which makes the remote calls transparent to your code.

To use these two variables, you must first authenticate the client application, and there are two ways to authenticate your application: Windows authentication (the application passes to the web service the credentials of the user who is executing the application) and Basic authentication (the application passes to the web service the username and password of an account). The first method is simpler and safer, as you don't have to store passwords anywhere; you simply execute the following statement, which uses the current user's credentials to connect to the Report Server:

```
RS.Credentials = System.Net.CredentialCache.DefaultCredentials
```

To use the Basic authentication method, you must create a new instance of the `System.Net.NetworkCredential` class, passing the current user's name and password in the instance's constructor, as in the following statement:

```
RS.Credentials = New System.Net.NetworkCredential(➢
              "username", "password", "domain")
```

You should not store any credentials in your application. Either prompt users for their name and passwords, or store them in a file or local database in encrypted form. It's recommended that you use Windows authentication, unless the end user can't have a Windows account (most likely a remote user).

The authentication method you'll use must be supported by the security settings of the Report Server's virtual directory. Make sure that IIS uses the appropriate authentication method for the Report Server's virtual directory. If the application isn't properly authenticated by IIS, the user will receive the HTTP 401 error: Access Denied. You can also implement a custom authentication scheme with the Reporting Services authentication extensions, a topic discussed in Chapter 14.

Calling Web Service Methods

Calling the methods of a proxy is identical to calling a method in any class of the `.NET` Framework. All web services expose public methods, which you call with the usual notation: *`className.methodName`*. Of course, you must supply the proper arguments, if needed. The following code shows how to call the `ListChildren` method of the `ReportingService2005` proxy class. The code is used to make a recursive call to the web service that returns an array of `CatalogItem` objects with a list of all items in the Report Server's root folder:

```
Dim rs As New ReportingService2005()
rs.Credentials = System.Net.CredentialCache.DefaultCredentials
Dim items As CatalogItem() = rs.ListChildren("/", True)
```

Now you can iterate through the elements of the items array and retrieve the names and basic properties of each item in the Report Server's root folder.

Connecting to Another Server

Each web service has a base URL, which is specified when you create the proxy. You can change the base URL from within your code at will by setting the `Url` property of the proxy. You can use this technique to copy a report from one Report Server to another (as long as the default credentials allow you to access both servers). The following code segment does exactly that:

```
Dim rs As New ReportingService2005()
rs.Credentials = System.Net.CredentialCache.DefaultCredentials
rs.Url = "http://Server1/ReportServer/ReportingService2005.asmx"
Dim reportName As String = "/SampleReports/AWCustomers"
Dim reportDefinition As Byte() = Nothing
Try
   reportDefinition = rs.GetReportDefinition(reportName)
   rs.Url = ➢
      "http://Server2/ReportServer/ReportingService2005.asmx"
   rs.CreateReport("Copy of AWCustomers", "/", False, ➢
      reportDefinition, Nothing)
```

```
Catch e As SoapException
   Console.WriteLine(e.Detail.InnerXml.ToString())
End Try
```

Like any procedure, web service methods have optional parameters. Even if you want to omit an optional parameter, you can't skip it. You must pass the `Nothing` values to it (which is the VB equivalent of the `Null` value).

Now that you have created the two proxies to the classes exposed by the Report Server web service, you can explore the object model they expose to familiarize yourself with the functionality these services expose. Right-click the proxy to the ReportingService2005 in the Solution Explorer and select View In Object Browser from the context menu. The Object Browser window opens, where you can view the members of the class, as shown in Figure 11.4.

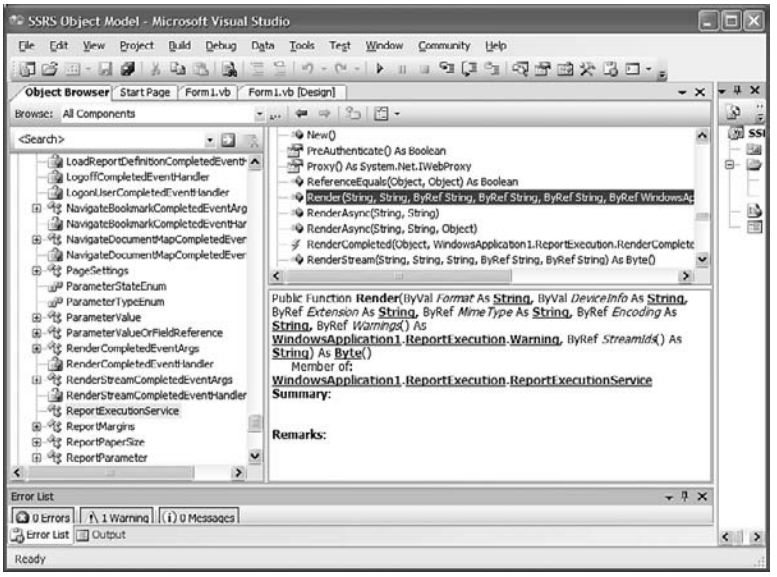


FIGURE 11.4
Viewing the object model of the ReportingService2005 web service

In the right pane you see a list of the classes that make up the ReportExecution web service. Locate the item `ReportExecutionService` to see the methods it exposes in the upper pane on the left. In this pane locate the `Render` item, which is the method that actually renders a report. You'll see the syntax of the method and an explanation of its arguments, as shown in Figure 11.4. This is not a recommendation that you try to figure out how to use the web service by looking up its members, but once you get a good idea of the service's functionality, you'll be able to locate the items you need each time.

Now we're ready to discuss some of the most common and most useful members of the web service. Let's break them down as management and execution methods: the management methods are exposed by the `ReportingService2005` class, and the execution methods are exposed by the `ReportExecution2005` class. For a complete list of the available methods, please see the product documentation. This chapter's goal is to familiarize you with the basic functionality of the two classes, which is in effect the functionality of the Report Server itself, and give you a few practical code samples, which you can use in your applications. Once you've familiarized yourself with the

basic operations of the two classes of the web service, you'll be able to research them further in the documentation.

Selected Management Methods

The `ReportingService2005` class of the Report Server web service exposes numerous methods for managing the items on the Report Server. The most important methods of the `ReportingService2005` class are listed in this section.

CreateFolder

The `CreateFolder` method adds a new folder to the Report Server database. Its syntax is

```
CreateFolder(FolderName, ParentFolderName, properties)
```

The method's first argument is the name of the new folder. The second argument is the name of the parent folder, and you can pass a set of properties with the last argument to the method. This argument is a collection of value/name pairs. You can assign arbitrary properties to a folder and use them later from within your code as needed. Notice that you can't set the usual folder properties, such as `Read-Only`, `Archive`, and so on. To create a new folder and assign to it the custom property `Department`, use the following statement to build the argument that specifies the custom property:

```
Dim newProperty As New [Property]()
newProperty.Name = "Department"
newProperty.Value = "Accounting"
Dim props(0) As [Property]
properties(0) = newProperty
```

Then you can pass the properties array to the `CreateFolder` method:

```
rs.CreateFolder("Accounting", "Departments", properties)
```

The method should be called from within a structured exception handler as shown next, because it may fail to create the new folder should a folder by the same name exist already, or should the authenticated user have no rights to create new items on the Report Server:

```
Try
   rs.CreateFolder("Acccounting", "Departments", properties)
Catch e As SoapException
   Console.WriteLine(e.Detail.InnerXml)
End Try
```

Note that we use the exception's `Detail.InnerXml` property to retrieve the description of the error returned by the Report Server. To delete or move an existing folder, use the `DeleteItem` and `MoveItem` methods, respectively, passing the name of the folder to be deleted as an argument. While there are specific methods for creating new items, depending on the type of item you're creating, all items are deleted and moved around with the same methods.

GetReportDefinition

The `GetReportDefinition` method returns the report definition of a report. Its syntax is

```
GetReportDefinition(ReportName) As Byte()
```

`GetReportDefinition` returns a byte array, which you can use with the `CreateReport` method, or convert it to an XML document in code and process it further. For an example of the usage of the `GetReportDefinition` method, see the following section on the `CreateReport` method.

CreateReport

The `CreateReport` method adds a new report to the Report Server. It accepts several arguments, and its syntax is

```
CreateReport (Report As String, Parent As String, ➢
              Overwrite As Boolean, Definition As Byte(),➢
              Properties As Property()) As Warning()
```

The `Definition` argument is an array of bytes that contains the RDL description of the report (the file generated by the visual tools of the Report Designer that contains the report's layout in the Report Definition Language). The `Report` argument is the report's name, and the `Parent` argument is the name of the parent folder of the report. The `Definition` argument is an array of bytes, and it's an XML document that describes the report. The last argument is an array of custom properties, and it's a collection of name/value pairs.

You're not supposed to create this document from scratch in your code, because it's not a trivial task, not to mention that you should make the most of a RAD environment like the Business Intelligence Visual Studio; you will most likely start with an existing report definition, modify it, and then add it to the Report Server's database. To grab the definition of an existing report, use the `GetReportDefinition` method. The following code segment reads the definition of the Report1 report and creates an identical report with a different name. This is how you can transfer reports from your development PC to a production server, or from a restricted area of the Report Server to a folder that can be accessed by end users.

```
Dim RS As New ReportService.ReportingService2005
RS.Credentials = System.Net.CredentialCache.DefaultCredentials
Dim bytes() As Byte = RS.GetReportDefinition( ➢
                          "/Report Project1/Report1")
TextBox1.Clear()
TextBox1.AppendText(System.Text.Encoding.UTF8.GetString(bytes))
RS.CreateReport("Report2", "/", False, bytes, Nothing)
```

The definition of the report is displayed in a TextBox control, and then the same byte array is used to create an identical report. You can use this code to copy a report definition from one server to another. If you execute these statements, you'll be able to see the XML description of the report. Here's the beginning of a report definition:

```
<Language>en-US</Language>
<LeftMargin>2.5cm</LeftMargin>
```

```
<RightMargin>2.5cm</RightMargin>
<TopMargin>2.5cm</TopMargin>
<BottomMargin>2.5cm</BottomMargin>
<rd:GridSpacing>0.25cm</rd:GridSpacing>
<PageWidth>21cm</PageWidth>
<PageHeight>29.7cm</PageHeight>
<DataSets>
  <DataSet Name="DataSource1">
    <Fields>
      <Field Name="OccupationId">
        <DataField>OccupationId</DataField>
        <rd:TypeName>System.Int64</rd:TypeName>
      </Field>
      <Field Name="Occupation">
        <DataField>Occupation</DataField>
        <rd:TypeName>System.String</rd:TypeName>
```

Further down you will find the query that populates the data set:

```
<DataSourceName>DataSource1</DataSourceName>
<CommandType>Text</CommandType>
<CommandText>SELECT * FROM Occupations</CommandText>
```

As you can see, it's easy to edit the query and add an `ORDER BY` clause, or a `TOP` clause without actually recompiling and redeploying the project. Of course, you can't change the list of selected fields without changing the report itself. You can use the `XMLStream` class to read the report definition in your code and hack it accordingly. In Chapter 13 you'll learn how to create RDL files entirely from within your code. Again, this is not the most practical approach in the world, but if you have several streamlined reports you can write a code generator and reuse it to create a bunch of similar reports. You will first create a template report and copy the main sections of the XML code that the visual tools will emit.

ListChildren

The `ListChildren` method returns a list of Report Server database items that are located under the specified item. To retrieve the items under the root folder, specify the string `/` as the argument to the method. To retrieve the items under the folder `Budget`, specify the string `"/Budget"`as an argument to the method. The `ListChildren` method returns an array of `CatalogItem` objects, and these objects can be data sources, folders, linked reports, reports, and resources. The following code segment displays the names of all objects in the root folder of the Report Server:

```
Dim items() As ReportService.CatalogItem
items = RS.ListChildren("/", True)
Dim item As ReportService.CatalogItem
For Each item In items
    Select Case item.Type
        Case ItemTypeEnum.DataSource
            Debug.WriteLine("DATASOURCE: " & item.Name)
        Case ItemTypeEnum.Folder
            Debug.WriteLine("FOLDER: " & item.Name)
```

```
        Case ItemTypeEnum.LinkedReport
            Debug.WriteLine("LINKED REPORTS: " & item.Name)
        Case ItemTypeEnum.Report
            Debug.WriteLine("REPORT: " & item.Name)
        Case ItemTypeEnum.Resource
            Debug.WriteLine("RESOURCE: " & item.Name)
    End Select
Next item
```

The output generated by this code segment looks like this:

```
DATASOURCE: AdventureWorks
FOLDER: Budget
FOLDER: Data Sources
FOLDER: Report Project1
REPORT: Sales Reports
REPORT: HR Reports
```

ListTasks

The `ListTasks` method returns the names and descriptions of system tasks. While you can retrieve the predefined tasks, there's no method to create new ones, because tasks can't be customized. To assign tasks to a role, see the description of the `SetRoleProperties` method, later in this section.

CreateRole

The `CreateRole` method adds a new role to the report server database. Its syntax is:

```
CreateRole (Name As String, Description As String, ➢
            Tasks As Task())
```

The first argument is the name of the role, the second argument is its description, and the last argument is an array of `Task` objects that represents the tasks to be assigned to the role. Notice that tasks are identified by their names, so you must write a `Select` statement to select the appropriate tasks for the role, as shown in the following code segment:

```
Dim reportName As String = "AWCustomers"
Dim reportDescription As String = "View list of customers"
Dim newTasks(2) As Task
newTasks(0) = New Task()
newTasks(1) = New Task()
newTasks(2) = New Task()
Dim allTasks As Task() = RS.ListTasks()
Dim tsk As Task
    For Each tsk In allTasks
        If tsk.Name = "View reports" Then
            newTasks(0) = tsk
        Else
            If tsk.Name = "View folders" Then
                newTasks(1) = tsk
            Else
```

```
                If tsk.Name = "View resources" Then
                    newTasks(2) = tsk
                End If
            End If
        End If
    Next
RS.CreateRole(reportName, reportDescription, newTasks)
```

DeleteRole

Use the `DeleteRole` method to delete a role from the Report Server. This method accepts a single argument, which is the name of the role to be deleted from the server.

GetPermissions

The `GetPermissions` method returns the user permissions that are associated with a particular item in the Report Server database:

```
Dim permissions() As String
permissions = RS.GetPermissions("/AWCustomers/AWCustomersPerCountry")
Dim permission As String
For Each permission In permissions
     Debug.WriteLine(permission)
Next
```

GetRoleProperties

The `GetRoleProperties` method returns an array of tasks that are associated with a specific role. Its first argument is the role's name, and the second argument, which is passed by reference, is set by the method to the description of the role:

```
GetRoleProperties (ByVal Name As String, ➢
                   ByRef Description As String) As Task()
```

GetSystemPermissions

The `GetSystemPermissions` method returns the user's system permissions. This method doesn't accept any arguments; it returns the system permissions of the authenticated user (the user who established the connection to the Report Server).

ListRoles

The `ListRoles` method returns the names and descriptions of roles that are managed by the Report Server. The method accepts a single argument, which is a member of the `SecurityScopeEnum` enumeration (*`All`*, *`Catalog`*, *`Model`*, *`System`*) and specifies the roles to be listed.

SetRoleProperties

The `SetRoleProperties` method sets custom properties for a specific role. Its syntax is:

```
SetRoleProperties (Name As String, Description As String, ➢
                   Tasks As Task())
```

The first argument is the name of the role to which we want to assign one or more tasks, the second argument is a description of the role (this is optional), and the last argument is an array of `Task` objects you want to assign to the role. The `SetRoleProperties` method is used in code much like the `CreateRole` method: first you create an array of predefined tasks, and then you call the method passing this array as an argument.

Selected Execution Methods

The `ReportingExecution2005` namespace exposes methods for executing and rendering reports. The two most important methods in this class are the `Render` method, which processes a specific report and renders it in a specific format, and the `RenderStream` method, which returns a stream associated with the rendered report. You can redirect this stream to a file, read it with your code, or pass it to a custom control that knows how to process it. Most reports accept parameters, which you can set from within your code with the `SetExecutionParameters` method.

GetReportParameters

The `GetReportParameters` method returns the report's parameters, and the `SetReportParameters` method sets the report's parameters.

```
GetReportParameters (➢
   Report As String, HistoryID As String, ➢
   ForRendering As Boolean, Values As ParameterValue(), ➢
   Credentials As DataSourceCredentials()) As ReportParameter()
```

SetExecutionParameters

The `SetExecutionParameters` method sets the parameter values for the current report. Its syntax is as follows:

```
SetExecutionParameters (Parameters As ParameterValue(), ➢
               ParameterLanguage As String) As ExecutionInfo
```

The first argument is an array of `ParameterValue` objects, which holds the values of all parameters. The second argument specifies the locale (a string like `en-us`) for the parameters and it must be specified. It's critical that you provide the correct locale if the parameters include dates or formatted numeric values. The method also returns a value, which is an `ExecutionInfo` object. This object contains information that will be used later by the `Render` method to generate the report. Assuming that the AWCustomers report accepts a single parameter, the `country` parameter, here's how you can supply the value of this parameter to the web service prior to calling the `Render` method:

1. Set up an array of `ParameterValue` objects. This array should contain exactly as many elements as there are parameters.
2. Instantiate each element of the array by setting it to a new instance of the `ParameterValue` class.
3. Set each parameter's name and value with the `Name` and `Value` properties, respectively.

The following code segment sets the `country` parameter of a report to the value `Australia`:

```
Dim RE As New ReportExecution.ReportExecutionService
RE.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim parameters (0) As ReportExecution.ParameterValue
parameters (0) = New ReportExecution.ParameterValue()
parameters (0).Name = "country"
parameters (0).Value = "Australia"

Dim execInfo As New ReportExecution.ExecutionInfo
execInfo = RE.LoadReport(reportPath, Nothing)
RE.SetExecutionParameters(parameters, "en-us")
```

Render

`Render` is the most important method of the `ReportExecution2005` class, and it renders a specific report in the specified format. The syntax of the `Render`method is:

```
Render (Format As String, DeviceInfo As String, ➢
        ByRef Extension As String, ByRef MimeType As String, ➢
        ByRef Encoding As String, ByRef Warnings As Warning(), ➢
        ByRef StreamIds As String()) As Byte()
```

The first argument is a string that identifies the format in which the report will be rendered; its value can be one of the following strings: `XML`, `NULL`, `CSV`, `IMAGE`, `PDF`, `HTML4.0`, `HTML3.2`, `MHTML`, `EXCEL`, or `HTMLOWC`. The `DeviceInfo` argument is an XML string that contains the device-specific information required by the format of the report specified by the first parameter. The `Extension` argument is a file extension corresponding to the output stream. The `MimeType` argument identifies the MIME (Multipurpose Internet Mail Extensions) type of the rendered report. The `Encoding` argument specifies the encoding used when the report is rendered. The `Warnings` argument is an array of `Warning` objects, and it returns all the warnings that occurred during the rendering of the report. The last argument, `StreamIds`, is an array of stream identifiers, which can be used later to render external resources associated with the report. You can use this array to break a long report into multiple files (if you want to render each page of the report as an image, for instance, you will need a separate stream for each image). Finally, the method's return value is a byte array with the contents of the report in the specified format. You can use the return value to display the report in the appropriate control, such as a WebBrowser control for HTML reports, a ListView control for CSV reports, and so on, or to process the data with a custom application.

Let's write a procedure to render a report using the `ReportExecution` class of the Report Server web service. The report we'll render is the PriceList report, which returns a price list from the AdventureWorks database. The report, in HTML format, is shown in Figure 11.5. In our code we'll retrieve the same data as a comma-separated list (with line breaks between records) and as XML.

The following code shows you how to render a report in just about any format. The report's format is specified with the *`format`* variable, which is set to `CSV`. After calling the `Render` method, the code converts the array of bytes returned by the method into a string by calling the `GetString`

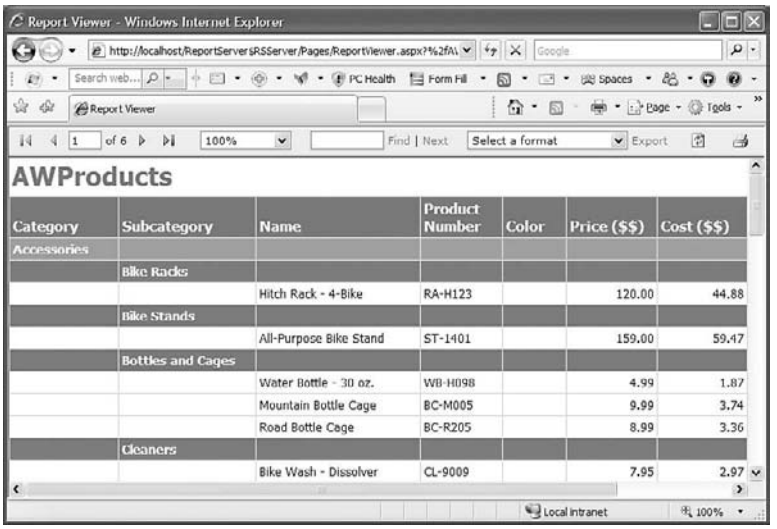


FIGURE 11.5
A simple price list for the AdventureWorks database

method of the `System.Text.Encoding.Unicode` class. This string is dumped on a TextBox control, from where you can copy it to the Clipboard and reuse it as you see fit.

```
Dim RE As New ReportExecution.ReportExecutionService
RE.Credentials = System.Net.CredentialCache.DefaultCredentials

Dim result As Byte() = Nothing
Dim reportPath As String = "/AWProducts/PriceList"
Dim format As String = "CSV"
Dim historyID As String = Nothing
Dim devInfo As String = ➢
          "<DeviceInfo><Toolbar>False</Toolbar></DeviceInfo>"

Dim warnings As ReportExecution.Warning() = Nothing
Dim streamIDs As String() = Nothing

Dim execInfo As New ReportExecution.ExecutionInfo
execInfo = RE.LoadReport(reportPath, historyID)

Try
    result = RE.Render(format, devInfo, "",  ➢
                       "", "", warnings, streamIDs)
Catch ex As System.Web.Services.Protocols.SoapException
    Console.WriteLine (ex.Detail.OuterXml)
End Try

Try
    TextBox1.Text = ➢
                System.Text.Encoding.Unicode.GetString(result)
```

```
Catch ex As Exception
    Console.WriteLine (ex.Message)
End Try
```

Place the preceding statements into a button's `Click` event handler and run the application. If you exclude the variable declarations and the structured exception handler, the code that actually renders the report is quite short. And it wouldn't make much sense otherwise; the report resides on the Report Server and all you have to do is instruct the web service to return the data that it would use to build the HTML page had you opened the same report from within your browser. The two necessary steps are the loading of the report (the `LoadReport` method) and its subsequent rendering (the `Render` method). You'll have to include some additional statements for reports that accept parameters, as you have seen earlier in this section.

The program will generate the following price list and display it in a TextBox control, which you must also place on the main form:

```
Category,Subcategory,Name,ProductNumber,Color,ListPrice,StandardCost
Accessories,Bike Racks,Hitch Rack - 4-Bike,RA-H123,,$120.00,$44.88
Accessories,Bike Stands,All-Purpose Bike Stand,ST-1401,,$159.00,$59.47
Accessories,Bottles and Cages,Water Bottle - 30 oz.,WB-H098,,$4.99,$1.87
Accessories,Bottles and Cages,Mountain Bottle Cage,BC-M005,,$9.99,$3.74
Bikes,Mountain Bikes,"Mountain-400-W Silver, 42",BK-M38S-42,Silver,$769.49,$419.78
```

I've included a line at the end that contains a comma in the product's name field to show you that the entire string was enclosed in double quotes. But this is the CSV format, and the application that will process the data should know how to read these values.

To save the report's output to a file, just replace the statement that populates the TextBox control with the following statements:

```
Dim stm As IO.FileStream = ➢
              IO.File.Create("report.html", result.Length)
strm.Write(result, 0, result.Length)
strm.Close()
```

To view the same data in XML format, just change the *`format`* variable from `CSV` to `XML` and run the code again. You must also change `encoding` from `Unicode` to `ASCII` (change the statement that populates the TextBox1 control). Copy the result on the TextBox control, start an XML editor such as XML Notepad 2007, and paste the data. XML Notepad 2007 is a free XML editor, which you can download from MSDN. Go to `http://msdn.microsoft.com/XML/XMLDownloads/default.aspx` and look for the `XML Notepad 2007` string in the Tools section. The editor will figure out the structure of the data and will display it as shown in Figure 11.6. You can navigate to any node and view the data. If you're familiar with XML you already know how to apply a style sheet to the data and display it in the browser or process it from within an application. As it turns out, Reporting Services is a flexible tool as it's not limited to rendering reports in a browser. You can retrieve raw data from a report and transform it, or process it in any way that makes sense for your business needs. For example, you can use XSLT transformations to present the same report differently to different groups of users. You can hide columns for users with fewer privileges, display totals only, and so on.

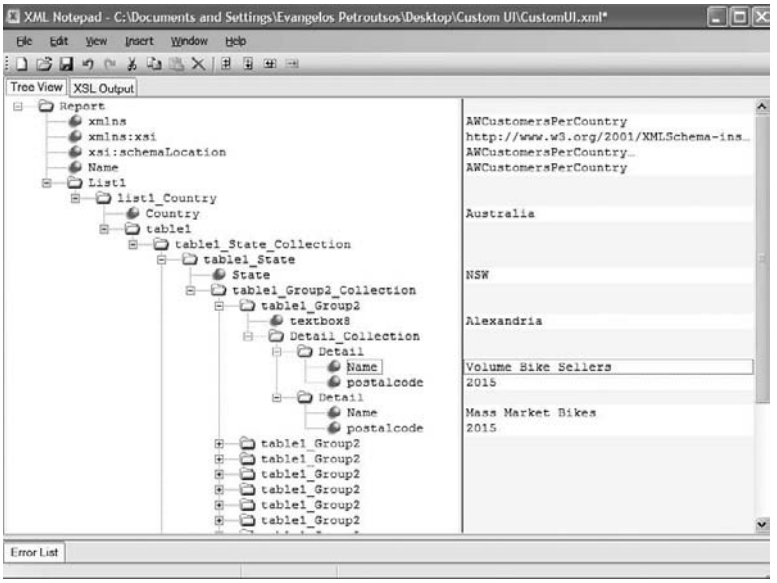


FIGURE 11.6
Viewing an XML report with XML Notepad

Real World Scenario

CASE STUDY: MANAGER REQUESTS

I recently set up a reporting system for a large music store to augment an Enterprise Resource Planning (ERP) application. Users requested a lot of similar reports, and I had no problem implementing them and organizing them into a folder structure and then providing shortcuts for the various report categories on the users' desktops. Once the manager and chief accountant, who had access to all reports, started using the reporting system, they requested that certain summary reports be incorporated into the main application's menu. Instead of using the ReportViewer control to render reports in a Windows form (which is no trivial task), I decided to write a small application that retrieved reports in HTML format, saved them to a temporary file, and then dumped this file into a WebBrowser control. The developers of the ERP application provided a hook to their menu from where I could call my own EXE application, and I was able to display any of the available reports in a Windows form that blended well with the rest of the application. When they decide to upgrade the system, I'll probably do something similar using XML files and XSLT transformations. This simple idea worked because the reports did not request any parameter value from the user, and I could pass the few parameters (department ID, range of dates, and so on) to the Report Server through the `SetExecutionParameters` method from within my code. This approach also worked because the reports they were interested in were short summary reports, nothing like the regular endless sheets users had to work with.

Specifying Device Information Settings

Let's focus a little on the `DeviceInfo` argument. This argument is an XML string that contains information required by the rendering extension specified by the `Format` parameter of the `Render`

method, and it's different for different formats. For a complete list of the elements you can use in defining device settings, see the section ''Device Information Settings'' in the Reporting Services documentation. Here are the basic device settings for HTML output (the default):

The HTML device settings are:

DocMap Determines whether the report's document map is visible or not.

HTMLFragment Determines whether the output will be rendered as an HTML fragment (if `True`) or as a full HTML document. An HTML fragment omits the `HTML` and `BODY` elements and contains the report in a `TABLE` element.

Parameters Determines whether or not the parameter prompt area of the toolbar is visible.

Toolbar Determines whether or not the toolbar is visible (the default value is `True`). If the value is `False`, the entire HTML report is rendered as a single HTML page, because the navigation tools, which appear on the toolbar, are not present.

Zoom The report's zoom value. Its value can be an integer (which is treated as a percentage) or one of the string values: `Page Width` or `Whole Page` (the same values that appear in the toolbar's Zoom ComboBox control).

Next we'll look at the CSV device settings. If you chose to retrieve the data as a CSV list, you can set one or more of the following elements:

Encoding The encoding of the output. Its value can be one of the strings `ASCII`, `UTF-7`, `UTF-8`, or `Unicode`.

NoHeader Set it to `True` to suppress the header row.

RecordDelimiter Set this element to specify a delimiter other than the default one (which is `<cr><lf>`).

''Silent'' Reporting

You can write Windows (or even console) applications that contact the Report Server and request reports in CVS or Excel format (the device settings for Excel worksheets are discussed shortly). You can use the appropriate CSV settings to retrieve the report's data in the desired format and then process them from within your application. The application doesn't have to provide a user interface and the reports aren't rendered at all; it can execute one or more reports against the server and store the results in local files, or process them further.

Now we'll look at XML device settings. If the output format is XML, you can use the following elements:

XSLT Use this element to specify an XSLT transformation to be applied to the data.

Indented Set this element to `True` to generate a properly indented XML file.

Encoding Same as with the CSV format.

OmitSchema Omits the schema from the output.

OmitFormulas Indicates whether to omit formulas from the rendered report. The default value is `False`.

Customizing Reports Outside Reporting Services

The Report Server's capability to return a report's data in XML format is an interesting feature, because it allows you to customize completely the appearance of the report through an XSLT transformation. In effect, you can customize an existing report without touching the RDL file; just retrieve the report in XML format and apply an XSLT transformation to create an HTML page that's totally different from the one returned by the Report Server. You can even exclude certain columns of sections of the report depending on the intended recipient.

If the output format is Excel, you can use the following elements:

`RemoveSpace` Set it to `True` to omit rows or columns that contain no data and are smaller than the given size, which is specified as a numeric value followed by the `in` string (for inches). If set, any rows that contain no report items will also be removed. The default value for the `RemoveSpace` setting is `0.125in`.

`SimplePageHeaders` Determines whether the header row will appear on top of each page. If set to `False`, the page header is rendered only on the first row of the worksheet.

Now we'll look at the Image device settings, and there are quite a few options when you're using the Image format:

`DpiX`, `DpiY` The horizontal and vertical resolution of the image. The default value is 96.

`PageHeight`, `PageWidth` The page's height and width in inches, specified by a numeric value followed by `in`. Use these elements to break the report into reasonably sized chunks.

`StartPage`, `EndPage` The first and last page of the report.

`MarginBottom`, `MarginLeft`, `MarginRight`, `MarginTop` The image's margins. They're expressed in inches and must be followed by `in` (`1in`, for example).

`OutputFormat` This element specifies the image type and its value can be one of the following strings: `BMP`, `EMF`, `GIF`, `JPEG`, `PNG`, or `TIFF`.

Strings in XML need not be enclosed in quotes. Here are a few examples of valid `DeviceInfo` arguments. Note that all `DeviceInfo` definitions must be enclosed in a <DeviceInfo> element and they're case sensitive:

```
<DeviceInfo><Toolbar>False</Toolbar>
            <OutputFormat>HTML4.0</OutputFormat>
</DeviceInfo>

<DeviceInfo><Toolbar>False</Toolbar>
       <StartPage>3</StartPage>
       <OutputFormat>EMF</OutputFormat>
</DeviceInfo>

<DeviceInfo><NoHeader>True</NoHeader>
       <OutputFormat>CSV</OutputFormat>
</DeviceInfo>
```

The Bottom Line

Integrate reports into Windows and web applications.

Master It Can you bypass the Report Server's interface and request individual reports at will?

Access reports with a URL.

Master It Any report on the Report Server has its own URL, which is displayed in the browser's address box every time you click the report's link in the main page.

Pass commands to the Report Server through a URL.

Master It What kind of operations can you perform against Report Server from within your browser besides viewing reports?

Manage reports with the `rs` command-line utility.

Master It The Report Server comes with its own web-based management tools. Can you automate specific operations on the Report Server?

Use the Report Server web service.

Master It The Report Server web service exposes a rich object model that allows you to access all of the functionality of Reporting Services from within your code. When should you use the web service?

Specify device information settings.

Master It How do you specify the settings of the device on which the report will be rendered?

Chapter 12

Extending Reporting Services

Reporting Services was designed from the ground up for extendibility, and you can create custom extensions for all major components of it. You can extend the data processing capabilities of the Report Server by implementing a data processing extension component that retrieves data from files, the filesystem, or any other source that's not supported by Reporting Services out of the box. Likewise, you can create a custom security extension to authenticate and authorize users. Once this component is installed, it takes over the default security, which is based on Windows accounts, and users are validated from within your custom code. Another type of extension is the delivery extension. This extension allows you to implement custom delivery mechanisms for sending report notifications to users.

Extensions are expected to duplicate the functionality of the original component, so that they can be used seamlessly in the design environment. The data processing extensions, for example, should be able to retrieve some data from a data source (you could even generate the data on the fly). To create a data processing extension, you must implement certain components that the Report Server expects to be there, such as a method to connect to a database; execute a command or stored procedure to select data; and read the data returned by the server. To implement a security extension, you must provide the methods to authenticate and authorize the user. Security extensions are quite useful, especially if the report requests come from users without a Windows account, or if you want to perform a custom authentication and authorization scheme from within your application.

This chapter is an overview of the extension capabilities of Reporting Services. Microsoft ships a few sample applications with the product, which you can find in the `Samples` folder under SQL Server's installation folder. The default location of the samples is `C:\Program Files\Microsoft SQL Server\90\Samples`. If you haven't installed the samples, you can run the installation program again and check the option for installing the samples. This chapter is based on the samples, and I'll focus on two extension types: security extensions and data processing extensions.

In this chapter you will:

- Explore the extension types
- Build security extensions
- Deploy security extensions
- Build data-processing extensions
- Deploy data-processing extensions

Understanding Extension Types

Reporting Services supports custom extensions that add functionality to both the Report Server and Report Designer. The most prominent type of extension is the data processing extension, which allows you to integrate nontraditional data sources into your reports. You can develop a custom data-processing extension to bring in your report data from files, even live data from data-acquisition devices. The data processing sample that comes with the product shows how to retrieve information from your computer's filesystem (folders and files along with their attributes) and use them to design and view reports. It's not the most practical extension, but it's a sample that can be used as is for demonstration purposes.

Another extension type is the delivery extension. With a custom delivery extension, you can use your own delivery mechanisms to send report notifications to users. You could write a delivery extension to send reports to a specific printer or fax reports to their recipients. The PrinterDelivery sample shows how to deliver a report directly to a printer.

Rendering extensions allow you to create reports in totally custom formats. Out of the box, Report Server can render reports as HTML pages, comma-separated value (CSV) files, images (TIFF, GIF, JPG), PDF files, and XML documents. If you need to format your data in a custom format, you can write your own custom rendering extension and deploy it to the Report Server.

The last type of extension, security extensions, enables you to implement custom security. You should implement a security extension if you want to authenticate and authorize users without Windows accounts. You will probably need to implement a security extension if you want to incorporate the functionality of Reporting Services in a custom application with its own user validation. If your application maintains a list of users and their privileges (application-related privileges that do not match the account's rights), you will probably implement a security extension that enables Report Server to authenticate and authorize users based on your application's validation schema. The FormsAuthentication sample project shows how to implement a custom security that relies on a table with usernames and passwords to authenticate users.

Let's start our exploration of Reporting Services extensions with the process of designing and deploying a custom security extension.

Building a Security Extension

If you need to design a custom extension for Reporting Services, it's likely that you will design a security extension to provide custom techniques for authenticating and authorizing users, so we'll explore this type of extension in detail. The code of the application is too lengthy to be printed in this book, but I'll base the discussion on one of the sample extension applications that come with Reporting Services. It's the VB version of the FormsAuthentication sample project. You can open this project to experiment with the programming techniques presented in this chapter.

Extensions are implemented as class libraries. The extension is usually implemented as multiple classes that are packaged in the same or separate DLL files, all under a common namespace.

1. Start Visual Studio and create a class library project, the CustomSecurity project. Visual Studio will create a new project and add an empty class to it, the `Class1.vb` class.

2. Right-click the project's name in Solution Explorer and select Add Reference from the context menu. In the Add Reference dialog box that appears, locate the Reporting Services Interfaces component in the `.NET` tab, select it, and click OK to add a reference to the interfaces of Reporting Services to your project.

3. Open the project's Properties (select Project ➢ Properties) and set the application's Assembly and Root namespace. The sample project uses the name `Microsoft.Samples.ReportingServices.FormsAuthentication` for both. The namespace should be globally unique, because you can have multiple components that expose identically named members, or even identically named components. Each component's name is uniquely identified by prefixing its name by its namespace.

4. The sample project consists of several classes, all under a common namespace, which is defined in the project's properties. Now delete the class inserted by Visual Studio and add a new class to the project, the `AuthenticationExtension` class, which will implement the `IAuthenticationExtension` interface. Then insert the following `Imports` statements in the `AuthenticationExtension` class:

```
ImportsSystem
ImportsSystem.Data
ImportsSystem.Data.SqlClient
ImportsSystem.Security.Principal
ImportsSystem.Web
ImportsMicrosoft.ReportingServices.Interfaces
Imports System.Globalization
```

   These statements import the corresponding namespaces, so that you won't have to fully qualify the names of the components. You will insert the same statements in the other classes of the project.

The *IAuthenticationExtension* Interface

In this section we're going to build the `AuthenticationExtension` class to implement the `IAuthenticationExtension` interface; this class will provide the methods for authenticating users.

1. Insert the following statement below the class definition:

```
Implements IAuthenticationExtension
```

   `IAuthenticationExtension` is an interface that contains the definitions of the procedures you must implement in your code. The runtime environment of Reporting Services expects to find these procedures in order to make use of your custom security extension. As soon as you press Enter at the end of the line, the editor will insert the following definitions in the code window:

```
Public Sub SetConfiguration(➢
          ByVal configuration As String) _
          Implements IAuthenticationExtension.SetConfiguration
End Sub

Public ReadOnly Property LocalizedName() As String ➢
          Implements IAuthenticationExtension.LocalizedName
    Get
        Return Nothing
    End Get
End Property
```

```
Public Function LogonUser(➢
          ByVal userName As String, ByVal password As String, ➢
          ByVal authority As String) As Boolean ➢
       Implements IAuthenticationExtension.LogonUser
End Function

Public Sub GetUserInfo(ByRef userIdentity As IIdentity, ➢
                  ByRef userId As IntPtr) Implements ➢
                IAuthenticationExtension.GetUserInfo
       If Not (HttpContext.Current Is Nothing) AndAlso _
            Not (HttpContext.Current.User Is Nothing) Then
           userIdentity = HttpContext.Current.User.Identity
       Else
           userIdentity = Nothing
       End If
       ' initialize a pointer to the current user id to zero
       userId = IntPtr.Zero
   End Sub

Public Function IsValidPrincipalName(➢ _
             ByVal principalName As String) As Boolean ➢
             Implements IAuthenticationExtension.IsValidPrincipalName
   Return VerifyUser(principalName)
End Function

Public Shared Function VerifyUser(➢ _
         ByVal userName As String) As Boolean
End Function
```

I've removed the comments and attributes in front of the members for clarity, but you will see them in your editor.

2. The sample project's `LogonUser` method calls the `VerifyPassword` method of a helper class (the `AuthenticationUtilities` class), which connects to a database and retrieves user information from a table. You should edit the connection string accordingly for your system. Alternatively, you can use a really simple technique to authenticate the user, something like this:

```
If userName.Contains("SYBEX") Then
   Return True
Else
   Return False
End If
```

Obviously, this technique can't be used in a real application; I'm suggesting it here so that you won't have to design a new table with usernames and encrypted passwords. Designing a custom security extension requires thoughtful preparation on your part. The code of the `LogonUser` method can get as complicated as you wish. My goal is to show you how to

create a simple security extension, deploy it, and then use it with Reporting Services. The `AuthenticationExtension` class has been more or less implemented for you; let's look at the descriptions of the members you must implement in your class.

GetUserInfo Method

Reporting Services will call the `GetUserInfo` method to obtain the user's identity; it doesn't verify the user. The method's code reads the user's credentials from the session cookie. If there's no cookie with credential information, the method returns an empty identity and the user will be redirected to a logon page. After logging in, users will not be prompted again for the duration of the session.

LogonUser Method

The `LogonUser` method uses the credentials retrieved by `GetUserInfo` to actually verify the user. The code of the sample `LogonUser` method passes the user's name and password to the `VerifyPassword` method of the `AuthenticationUtilities` class. The `VerifyPassword` method can look up the name and password in a table with usernames, but it's a bad idea to keep passwords in your database, unless you encrypt them somehow. As you will see in the implementation of the `VerifyPassword` method, the code stores the password's hash code instead of the actual password.

The hash code is a unique string that's generated from a document (the password in our case) and a secret key. There are several algorithms for creating hash codes, including the MD5 (Message Digest algorithm 5) and variations of the SHA (Secure Hashing Algorithm), which are implemented in the `.NET` Framework. Hash codes have a length from 128 to 512 bytes, regardless of the size of the document's size, so in a way they are the document's signature. Although hash codes haven't been proven to be secure, each document produces a different hash code that uniquely identifies the document.

When new users are added, the code stores the hash code of their password, not the actual password, in the database. When the same user attempts to connect, the security extension hashes the user-supplied password with the same key used to create the hash code stored in the database and then it compares hash codes instead of passwords. If the hash codes match, we're quite confident that the passwords will also match. This simple trick allows you to avoid storing passwords in text format in your database. Even if someone has obtained the key used in hashing the password, they still can't retrieve the password, because it's not stored anywhere. Only the users know their password, and they must enter it in a web page at the client. An intruder who accesses your database won't be able to retrieve password information either. All you have to do is protect the key you use to hash the passwords.

IsValidPrincipalName Method

The `IsValidPrincipalName` method determines whether the user name is a valid one — a task that's performed by the `VerifyUser` method. Reporting Services calls the `IsValidPrincipalName` method when a new role assignment is created to validate the username of the new role assignment. If the username is valid, the new assignment is stored in the Reporting Services database.

In addition to the `AuthenticationExtension` class, which provides all authentication methods, the FormsAuthentication project contains a logon page, the `Logon.aspx` page. This is a simple page that prompts the user for a username and password and has two buttons: one for

registering new users and another one for logging on existing users. The code behind the logon button is shown here:

```
Private Sub ServerBtnLogon_Click(ByVal sender As Object,
                    ByVal e As System.EventArgs)  ➢
                    Handles BtnLogon.Click
    Dim passwordVerified As Boolean = False
    Try
        passwordVerified = AuthenticationUtilities. ➢
                  VerifyPassword(TxtUser.Text, TxtPwd.Text)
        If passwordVerified Then
                FormsAuthentication.RedirectFromLoginPage ➢
                                (TxtUser.Text, False)
        Else
            Response.Redirect("logon.aspx")
        End If
    Catch ex As Exception
        lblMessage.Text = ex.Message
        Return
    End Try
End Sub
```

The code calls the `VerifyPassword` method (the same one called from within the extensions code) to verify the user. Once the user has logged on, a cookie is created by the Report Server and maintained for the duration of the session. Every time the user requests an operation, such as viewing a report, deleting an item, and so on, Report Server calls the `CheckAccess` method of another class that implements the `IAuthorizationExtension` interface, passing the user's credentials as an argument. The class that handles authorization, named `AuthorizationExtension`, is discussed in the following section.

The *IAuthorizationExtension* Interface

The second major component of a security extension is the class that implements the `IAuthorizationExtension` interface. This interface, which provides the methods for authorizing users for various operations, is implemented in the sample project by the `Authorization` class. Add a new class to the project and name it `Authorization.vb`. The first statement in this class should be:

```
Implements IAuthorizationExtension
```

As soon as you press Enter at the end of the statement, the editor will insert the definitions of the members you must implement, as required by the interface.

The `IAuthorizationExtension` interface contains the definitions for a large number of methods, but you do not have to implement all of them. The most important method of the interface is the `CheckAccess` method, which is called every time a user attempts an operation on the Report Server. Moreover, the `CheckAccess` method is overloaded and you must provide the code for all forms of the method. Each overloaded form of the method is called under different circumstances

depending on the operation, and you may skip some implementations if you know that certain operations will not be performed. The operation is determined by the last argument of the method, the `requiredOperation` argument. When the security extension is in place and a user attempts to create a new subscription, the Report Server calls the `CheckAccess` method passing the value `requiredOperation.CreateSubscription` with the last argument. You can examine the value of the last argument in conjunction with the other arguments, and authorize the user to perform the operation or not.

This method is called all the time by the Report Server. When the user opens the home page, this method is called many times to determine the objects that the specific user is allowed to view or edit. So, the implementation of the `CheckAccess` method should be as efficient as you can make it. Here's the first overloaded form of the `CheckAccess` method:

```
Public Function CheckAccess(➢ _
   ByVal userName As String, ByVal userToken As System.IntPtr, ➢
   ByVal secDesc() As Byte, ByVal requiredOperation As ➢
   Microsoft.ReportingServices.Interfaces.CatalogOperation) ➢
   As Boolean
Implements Microsoft.ReportingServices.Interfaces.  ➢
           IAuthorizationExtension.CheckAccess
```

The *`userName`* argument is the name of the user making the request to the Report Server. The *`userToken`* argument is a handle to a Microsoft Windows account, which you can use to retrieve the user's rights. *`secDesc`* is the security descriptor for the item being accessed, and the last argument indicates the operation requested by the user.

The item's security descriptor is a collection of usernames (or group names) and catalog operation names in binary format, and here how it's used. First, it must be deserialized into a collection; this is usually done by a helper routine. The `DeserializeAcl` function accepts as argument the *`secDesc`* array of bytes and returns a collection (see Listing 12.1).

LISTING 12.1: Deserializing an Item's Security Description

```
Private Function DeserializeAcl(➢
                secDesc() As Byte) As AceCollection
   Dim bf As New BinaryFormatter()
   Dim sdStream As New MemoryStream(secDesc)
   Dim acl As AceCollection = ➢
            CType(bf.Deserialize(sdStream), AceCollection)
   Return acl
End Function
```

The `DeserializeAcl` function is called by the `CheckAccess` method. Then the `CheckAccess` method's code iterates through the items of the `AceCollection` collection. These items are `AceStruct` objects and they expose several properties. The two properties we're interested in are the `PrincipalName` (which contains usernames) and `CatalogOperations` (which

contains the operations for which the current user is authorized). If the operation specified by the last argument of the `CheckAccess` method belongs to the `CatalogOperations` list for the specific user, then we authorize the request by returning the True value from the `CheckAccess` method. If not, we return False. Listing 12.2 shows the implementation of the first overloaded form of the `CheckAccess` method.

LISTING 12.2: The *CheckAccess* Method for a Catalog Operation

```
Public Function CheckAccess(ByVal userName As String, ➢
        ByVal userToken As IntPtr, ➢
        ByVal secDesc() As Byte, ➢
        ByVal requiredOperation As CatalogOperation) As Boolean ➢
        Implements IAuthorizationExtension.CheckAccess
    Dim acl As AceCollection = DeserializeAcl(secDesc)
    Dim ace As AceStruct
    For Each ace In acl
        If userName = ace.PrincipalName Then
            Dim aclOperation As CatalogOperation
            For Each aclOperation In ace.CatalogOperations
                If aclOperation = requiredOperation Then
                    Return True
                End If
            Next aclOperation
        End If
    Next ace
    Return False
End Function
```

Notice that the code can't retrieve the collection item that corresponds to the current user; instead, it iterates through the entire collection and compares their `PrincipalName` property to the *`userName`* variable. If they match, the user's entry has been located. Then a nested loop goes through the `CatalogOperation` items of the current collection element. These items are the rights assigned to the specific user. If it locates the requested operation in this collection, it returns True. Otherwise, it returns False.

Another overloaded form of the `CheckAccess` method is used for an array of operations, such as when the user checks several items to delete. The last argument of this form of the method is an array of `CatalogOperation` objects and is defined as follows:

```
Function CheckAccess (userName As String, ➢
       userToken As IntPtr, secDesc As Byte(),➢
       requiredOperations As CatalogOperation()) As Boolean
```

The implementation of this form of the method calls the previous one, once for each of the specified operations, as shown in Listing 12.3.

LISTING 12.3: The *CheckAccess* Method for Multiple Catalog Operations

```
Public Function CheckAccess(➢
          userName As String, userToken As IntPtr, ➢
          secDesc() As Byte, ➢
          requiredOperations() As CatalogOperation) As Boolean
   Dim operation As CatalogOperation
   For Each operation In  requiredOperations
      If Not CheckAccess(userName, userToken, ➢
                         secDesc, operation) Then
         Return False
      End If
   Next operation
   Return True
End Function
```

Note that this is not a recursive function. The implementation of the `CheckAccess` function in Listing 12.3 does not call itself; it calls the overloaded form of the `CheckAccess` function shown in Listing 12.2 (the two overloaded forms differ in their last argument, which is a `CatalogItem` object for the function in Listing 12.2 and an array of `CatalogItem` objects for the function in Listing 12.3). For data source operations, the Report Server calls the following overloaded form of the method, whose last argument is of the `DataSourceOperation` type and represents an operation on a data source. `DataSourceOperation` is an enumeration whose members represent the operations you can perform on data sources, such as `Delete`, `ReadContent`, `UpdateContent`, and so on:

```
Public Overloads Function CheckAccess(➢
          ByVal userName As String, ByVal userToken As IntPtr, ➢
          ByVal secDesc() As Byte, ➢
          ByVal requiredOperation As DatasourceOperation) ➢
          As Boolean
```

You can look up the remaining overloaded forms of the `CheckAccess` method in the documentation and implement them according to the sample code.

Of course, you must insert custom code as needed. For example, you may prevent remote users from deleting certain items, regardless of the security settings of the Report Server. If you have a custom application with its own authorization settings, you may wish to access the related tables of your application to determine whether a user can view a report. Obviously, these items must be hidden on the Report Server; otherwise, users could bypass your authentication scheme.

The key to the custom security extension is the *`AceCollection`* variable that stores the current user's permissions for all operations. One of the methods of the `IAuthorizationExtension` interface you must implement is the `GetPermissions` method, which returns the current user's permissions as an array of strings. The `GetPermissions` method grants the administrator rights for all operations. For other users, it takes into consideration the *`secDesc`* argument, which contains the operations granted to the specified user by the administrator of the Report Server.

You should copy the implementation of this method from the FormsAuthentication project along with the `InitializeMaps` procedure and the local variable declarations and paste it in your project. Even better, you can copy all of the code in the sample project's `Authorization` class and modify the overloaded forms of the `CheckAccess` method accordingly.

Finally, you must implement the `SetConfiguration` method, which appears in Listing 12.4.

LISTING 12.4: The *SetConfiguration* Method

```
Public Overloads Sub SetConfiguration(➢
             ByVal configuration As String) ➢
      Implements IAuthorizationExtension.SetConfiguration
    Dim doc As New XmlDocument()
    doc.LoadXml(configuration)
    If doc.DocumentElement.Name = "AdminConfiguration" Then
        Dim child As XmlNode
        For Each child In doc.DocumentElement.ChildNodes
            If child.Name = "UserName" Then
                m_adminUserName = child.InnerText
            Else
                Throw New Exception("Unrecognized element.")
            End If
        Next child
    Else
        Throw New Exception(➢
               "No AdminConfiguration element found."))
    End If
End Sub
```

This method retrieves the `AdminConfiguration` element of the `RSReportServer.config` file, which is an XML file, and stores the administrator's username in the *`m_adminUserName`* variable. The `CheckAccess` methods should give this user full rights to all items, because when you switch to the custom security extension for the first time, there are no role assignments and only an administrator can create them. To grant all rights to the administrator specified in the configuration file, insert the following statement in each of the overloaded forms of the `CheckAccess` method:

```
If userName = m_adminUserName Then Return True
```

This statement should appear before the code that iterates through the security descriptors, of course. After deploying the security extension, you must log in as administrator and assign roles to the users.

Deploying a Custom Security Extension

Deploying the custom extension is not a trivial process, and Visual Studio doesn't provide any tools to automate the process. You must copy certain files to specific folders and edit a couple of configuration files. It's a process that involves a number of steps, and I'll go through them in detail.

The process described in this section should take place in a test environment, not a production server. I'm assuming that you're going to install the extension on the same machine that you

used to develop it, which has Reporting Services already installed. If something goes wrong in the installation process, you should be able to back out the changes and revert to the original installation. At the very least, you should first make copies of the configuration files you're editing so that you can undo your edits. If the installation succeeds, you can uninstall it later and revert to Windows integrated security.

1. Build the solution in Visual Studio to generate a DLL file that contains the extension's executable code. Use the sample application if you don't want to write a new one from scratch as explained earlier.
2. Switch to the `bin\Release` folder under the project's folder, where you will find the file `CustomSecurity.dll`. Select and copy the DLL file (right-click the selected file and choose Copy from the context menu).
3. Switch to the `bin` folder of Report Manager and paste the copied file there. The default location for this folder is: `C:\Program Files\Microsoft SQL Server\MSSQL\Reporting Services\ReportManager\bin`.
4. Paste the same DLL file in the `ReportServer\bin` folder. The default location of the `ReportServer` folder is on the same folder as the `ReportManager` folder.
5. Now copy the `UILogon.aspx` file from your project's `bin/Debug` folder into the `ReportManager\Pages` folder and the `Logon.aspx` file in the `ReportServer` folder.

Now you must modify a few configuration files following these steps:

6. Switch to the `ReportManager` folder and open the `rsmgrpolicy.config` file with a text editor. This file contains the configuration of Report Server's code access security. Find the element `MyComputer` and change the `permissionSetName` attribute from `Execution` to `FullTrust`.

```
<CodeGroup
   class="FirstMatchCodeGroup"
   version="1"
   PermissionSetName="Execution"
   Description="This code group grants MyComputer code
                Execution permission. ">
       <IMembershipCondition
           class="ZoneMembershipCondition"
           version="1"
           Zone="MyComputer" />
</CodeGroup>
```

   Save your changes to the `rsmgrpolicy.config` file.

7. Now open the `RSWebApplication.config` file with Notepad. This file contains configuration information about the Report Manager web configuration. Locate the `<UI>` element and add the following code:

```
<UI>
  <ReportServerUrl></ReportServerUrl>
  <ReportServerVirtualDirectory>ReportServer
  </ReportServerVirtualDirectory>
  <CustomAuthenticationUI>
    <LogginUrl>/Pages/UILogon.aspx>
    <UseSSL>False</UseSSL>
```

```
    </CustomAuthenticationUI>
  <ReportServerUrl>
     http://localhost/ReportServer</ReportServerUrl>>
</UI>
```

This configuration segment instructs Report Manager to redirect users that can't be authenticated to the UILogon page. Save your changes.

8. Open the `Web.config` file in the root folder of the Report Server's website with Notepad and change the value of the element `<identity impersonation>` from True to False (you don't want applications to log on by impersonating another user's credentials with your custom security extension):

```
<identity impersonate="False"/>
```

Save your changes to the web application configuration file.

9. Now switch to the folder `Reporting Services\ReportServer` where you will edit three more configuration files. Open the `RSReportServer.config` file and change the `Security` element as follows:

```
<Security>
  <Extension Name="Forms"
     TYpe="CustomSecurity.AuthorizationExtension,
           CustomSecurity">
    <Configuration>
      <AdminConfiguration>
        <UserName>SYBEX</UserName>
      </AdminConfiguration>
    </Configuration>
  </Extension>
</Security>
<Authentication
    <Extension Name = "Forms"
        Type="CustomSecurity.AuthenticationExtension,
              CustomSecurity"/>
    </ExtensionName>
</Authentication>
```

Now save the changes you made to the two files.

10. Next, open the file `rssrvpolicy.config` file, which contains configuration information for Report Server, and add the following new code group for the custom security assembly:

```
<CodeGroup>
  class="UnionCodeGroup"
  version="1"
  Name="SecurityExtensionCodeGroup"
  Description = "..."
  PermissionSetName = "Full Trust">
  <IMembershipCondition>
    class="UrlMembershipCondition"
```

```
        version="1"
        Url="C:\Program Files\Microsoft SQL
            Server\MSSQL\ReportingServices\ReportServer\
            bin\CustomSecurity.dll"/>
    </CodeGroup>
```

This code group assigns Full Trust rights to the custom extension (the `CustomSecurity.dll` file). This action will grant your executable (the DLL that implements the security extension) full trust rights, so the CLR will execute it. Executables without the proper rights simply won't be executed by the CLR. You may have to change the URL based on the location of the Reporting Services installation directory. Save the changes to the `rssrvpolicy.config` file.

11. Open the `Web.config` file in the Report Server's website's root folder for editing with Notepad. This is the configuration file for the Report Server web service, and we're going to change the default ASP authentication mode.

12. Locate the `<identity>` tag and change the `impersonate` attribute from true to false:

```
<identity impersonate="false"/>
```

13. Locate the `<authentication>` tag and change its `mode` attribute from `Windows` to `Forms`. You must also insert a `<forms>` tag with the login form:

```
<authentication mode="Forms">
  <forms loginURL="logon.aspx" name=sqlAuthCookie"
         timeout="60" path="/" />
</authentication>
```

14. Insert the following `<authorization>` tag right after the `<authentication>` tag:

```
<authorization>
  <deny users="?" />
</authorization>
```

Save the changes to the `Web.config` file.

15. Now we're going to configure IIS security. To do so, start the IIS management console. Select Administrative Tools in the Control Panel and double-click Internet Information Server. Expand the tree on the left pane under Default Web Site and locate the virtual folder for Reporting Services (it's the `Reports` folder).

16. Right-click the Reports item and select Properties from the context menu to open the Reports Properties dialog box. Switch to the Directory Security tab and click the Edit button in the Anonymous Access and Authentication Control section.

17. In the Authentication Methods dialog box that appears, check the Anonymous Access box. Next, specify an account and its password for users connecting to the web server without a Windows account. Finally, click OK twice to close the Reports Properties dialog box.

18. Now open the properties of the `ReportServer` virtual directory. In the ReportServer dialog box, switch to the Directory Security tab and enable anonymous access as before. Close the dialog boxes and exit the Internet Information Services management console.

These are the steps required to configure your custom security extension and integrate it with Reporting Services. For the extension to take effect, you must restart IIS. If you attempt to connect to Report Manager with your browser, you will see the custom logon page, where you must specify a username and a password to connect to Reporting Services. If you've used the SYBEX username as suggested in the implementation of the `LogonUser` method, you simply type this name and no password. Otherwise, you must create a table with valid usernames and provide the `LookupUser` stored procedure to verify usernames and passwords. Initially, the users in your database have no role assignments. You must connect as administrator (or give the SYBEX user full rights in the `CheckAccess` method) and log in with administrative privileges to assign roles to users.

If you have followed the preceding steps to make sure you can install a custom extension, you probably want to remove it now. To remove an extension, you must remove the <`Extension`> tag that corresponds to the security extension from the configuration files `RSReportServer.config` and `RSReportDesigner.config`. After you remove the configuration, the security extension is no longer available and Reporting Services will revert to the default Windows authentication and the previously configured roles.

Building a Data Processing Extension

To follow the process of building a data processing extension, we'll use the VB version of the FSIDataExtension Sample project that comes with the package. This extension uses the filesystem as the data source and allows you to create reports with folder and filenames and file attributes (such as file sizes and creation dates). The filesystem is not a particularly useful data source, but you can test the extension on any system without test data. As you will see, the extension imitates a `DataSet`. Once you understand the basics of implementing and deploying a data-processing extension, you can replace the sample with custom code that generates a `DataSet` based on data from any other data source.

The data processing extension should implement one or more of the following interfaces:

Required Interfaces The following interfaces should be implemented in the data processing extension class:

`IDbConnection` Represents a connection to a database server

`IDbCommand` Represents a query or stored procedure that will be executed against the database

`IDbTransaction` Represents a local transaction

`IDataParameter` Represents a parameter to be passed to the query or stored procedure or name/value pair that is passed to a command or query

`IDataParameterCollection` Represents a collection of all parameters relevant to a command or query

`IDataReader` Provides a method of reading a forward-only, read-only stream of data from your data source

Optional Interfaces The following interfaces are optional, and you should implement them depending on the functionality of your extension class:

`IDbConnectionExtension` Represents security and authentication properties for the connection

`IDbTransactionExtension` Represents transaction properties for the command

`IDbCommandAnalysis` Represents information for the command; also returns the names of the parameters used in the query

`IDataReaderExtension` Provides a method for reading the result set returned by the query

A data processing extension is the most complicated one, because you have to duplicate the functionality of a data provider. However, it's a straightforward process, especially if you're accustomed to programming with ADO.

The *IDbCommand* Interface

In the sample application, the class that implements the `IDbCommand` interface is called `FSICommand.vb`. This class implements the following members:

`CommandText` Property Implements the `IDbCommand.CommandText` interface

`CommandTimeout` Property Implements the `IDbCommand.CommandTimeout` interface

`CommandType` Property Implements the `IDbCommand.CommandType` interface

`Parameters` Property Implements the `IDbCommand.Parameters` interface and it's read-only

`Transaction` Property Implements the `IDbCommand.Transaction` interface

`Cancel` Method Implements the `IDbCommand.Cancel` interface

`CreateParameter` Method Implements the `IDbCommand.CreateParameter` interface and returns an `IDataParameter` object

`ExecuteReader` Method Implements the `IDbCommand.ExecuteReader` interface and returns an `IDataReader` object

`Dispose` Method Implements the `IDisposable.Dispose` interface

Typical Implementations

Implementing the preceding members is straightforward, and I've selected a few members to show you. The `CommandText` property represents the command that will be executed against the database; here's its implementation:

```
Property CommandText() As String ➢
         Implements IDbCommand.CommandText
    Get
        Return m_cmdText
    End Get
    Set(ByVal value As String)
        m_cmdText = value
    End Set
End Property
```

The `CreateParameter` method creates a new `IDataParameter` object:

```
Function CreateParameter() As IDataParameter ➢
         Implements IDbCommand.CreateParameter
    Return CType(New FsiDataParameter(), IDataParameter)
End Function
```

The FSIDataParameter is an instance of the FSIDataParameter class, which is discussed shortly. The ExecuteReader method (Listing 12.5) executes the command against the database and returns an IDataReader object, which will be used to retrieve the rows returned by the query. ADO.NET exposes an ExecuteReader method for retrieving the data returned by the database.

LISTING 12.5: Implementing the *ExecuteReader* Method

```
Function ExecuteReader(ByVal behavior As CommandBehavior) ➢
            As IDataReader ➢
            Implements IDbCommand.ExecuteReader
    Dim impersonationContext As ➢
            WindowsImpersonationContext = Nothing
    Dim reader As FsiDataReader = Nothing
    Try
        If Not (m_connection.ConnectionUser Is Nothing) Then
            impersonationContext = ➢
                m_connection.ConnectionUser.Impersonate()
        End If
        reader = New FsiDataReader(m_cmdText)
        reader.GetDirectory(m_cmdText)
    Finally
        If Not (impersonationContext Is Nothing) Then
            impersonationContext.Dispose()
        End If
    End Try
    Return reader
End Function
```

The *IDbConnection* Interface

In the sample application, the class that implements the IDbConnection interface is called FSIConnection.vb. This class implements the following members:

ConnectionString Property Sets or returns the connection string

ConnectionTimeout Property Returns the connection's timeout

State Property Returns the connection's state

BeginTransaction Method Initiates a new transaction

Open Method Opens the connection

Close Method Closes the connection

CreateCommand Method Creates a new Command object

IntegratedSecurity Property Reads or returns integrated security settings

UserName Property Sets the name of the user requesting the connection

Password Property Sets the password of the user requesting the connection

`Impersonate` Property Sets the impersonation information for the connection

`Dispose` Method Disposes of the connection

`ConnectionUser` Property Returns the windows identity of the user

Typical Implementations

The members of the `IDbConnection` interface are trivial, and you can look them up in the sample project. Only the `Open` method is of interest. In this method, you must call the `Open` method of a `Connection` object, which was instantiated in the class's `New` constructor:

```
Private CN As SqlClient.SqlConnection
Public Sub New(ByVal conn As String)
    CN.ConnectionString = conn
    CN.Open()
End Sub
```

The `State` property's code should return the `State` property of the `CN` object. Likewise, the `BeginTransaction` method should call the same `Connection` object's `BeginTransaction` method and return the `Transaction` object passed by the `CN.BeginTransaction` method. Notice that the `RollBack` and `Commit` methods are not members of the `IDbConnection` interface; you will find them in the `FSITransaction` class, which implements the `IDbTransaction` interface. For reporting purposes you'll rarely have to implement transactions, and you can skip the implementation of the `IDbTransaction` interface.

The *IDataReader* Interface

In the sample application, the class that implements the `IDataReader` interface is called `FSIDataReader.vb`. This class implements the following members:

`GetFieldType` Method Gets the `Type` information for a field specified by its index (ordinal value)

`GetName` Method Gets the name of a field specified by its index (ordinal value)

`GetOrdinal` Method Gets the index of a field specified by its name

`GetValue` Method Gets the value of a field specified by its index (ordinal value)

`Read` Method Advances the `IDataReader` to next row

`FieldCount` Property Returns the number of columns in the data reader

Typical Implementations

The implementation of all members of the `IDataReader` interface is trivial. The `FieldCount` property and the `GetName` method are listed here:

```
Public ReadOnly Property FieldCount() As Integer Implements IDataReader.FieldCount
    Get
        Return m_fieldCount
    End Get
End Property
```

```
Public Function GetName(ByVal fieldIndex As Integer) As String ➢
Implements IDataReader.GetName]
    Return m_names(fieldIndex)
End Function
```

The one method you must flesh out is `Read`, which is implemented with the following code in the sample application. All variables prefixed with the character `m` and an underscore are local variables declared outside the procedure.

```
Public Function Read() As Boolean Implements IDataReader.Read
    If Not (m_ie Is Nothing) Then
        Dim notEOF As Boolean = m_ie.MoveNext()
        If notEOF = True Then
            m_currentRow += 1
            If TypeOf m_fsi(m_currentRow) Is FileInfo Then
                Dim f As FileInfo = ➢
                     CType(m_fsi(m_currentRow), FileInfo)
                m_cols(0) = f.Name
                m_cols(1) = f.Length.ToString
                m_cols(2) = "File"
                m_cols(3) = f.CreationTime.ToString
            Else
                Dim d As DirectoryInfo = ➢
                      CType(m_fsi(m_currentRow), DirectoryInfo)
                m_cols(0) = d.Name
                m_cols(1) = "0"
                m_cols(2) = "Directory"
                m_cols(3) = d.CreationTime.ToString
            End If
        End If
        Return notEOF
    End If
    Return False
End Function
```

The *`m_ie`* variable is an Enumerator, which is initiated by the `GetDirectory` helper function, shown here:

```
Friend Sub GetDirectory(ByVal cmdText As String)
    ValidateCommandText(cmdText)

    m_dir = New DirectoryInfo(cmdText)
    m_fsi = m_dir.GetFileSystemInfos()

    m_currentRow = -1
    m_ie = m_fsi.GetEnumerator()
End Sub
```

As you realize, there's no command to be validated, but in a more complex situation you'd have to validate and execute the command.

The `Read` method goes through the data set's rows and creates the appropriate columns. There's no data set per se, but the code creates a new row on the fly for each folder or file entry. In an application that uses a `DataSet` object, you would return the columns and rows of the actual `DataSet`.

Each data processing extension you create for Reporting Services should have a unique namespace, like security extensions. Set the extension's namespace in the project's Properties pages, as you did for the security extension. A safe technique for assigning unique namespaces to your extensions is to use the company name. Microsoft, for example, uses the namespace `Microsoft.ReportingServices.DataProcessing` for the data processing extensions.

Deploying a Data Processing Interface

After writing your data processing extension, you must incorporate it in Reporting Services, a process that requires quite a few steps. As with security extensions, you must copy the DLL file(s) that implement the extension to specific folders and then edit the appropriate Reporting Services configuration files. The extension you want to deploy to the Report Server or Report Designer must be inserted with `<Extension>` tags in the configuration files: the `RSReportServer.config` file for the Report Server and the `RSReportDesigner.config` file for the Report Designer. The `<Extension>` tag recognizes the following attributes:

`Name` The extension's name, which must be unique in the configuration file

`Type` A string with the fully qualified namespace and the name of the assembly, separated a comma

`Visible` A True/False value indicating whether the data processing extension should be visible in user interfaces

Reporting Services use data processing extensions for retrieving and processing data when designing and rendering reports, and it expects to find the DLL files that implement the data processing extensions in specific folders. As with the security extension, you'll start by copying the extension's DLL file to a specific folder of the Reporting Services installation, and then you'll edit some configuration files.

1. Build your solution and then switch to the project's folder `bin\Release` subfolder. Copy the DLL file from this folder into the Report Server's `bin` folder. The default location of this folder is `C:\Program Files\Microsoft SQL Server\MSSQL\Reporting Services\ReportServer\bin`.
2. Then open the `RSReportServer.config` file with a text editor (this file is located in the `ReportServer` directory).
3. Locate the `< Data >` tag in the `RSReportServer.config` file and insert the following entry that identifies your extension:

```
<Extensions>
<Data>
<Extension Name="FSIDataExtension"
   Type="Microsoft.FSIDataExtension.FSIConnection,
         FSIDataExtension" />
</Data>
</Extensions>
```

   Save the changes in the `RSReportServer.config` file.

4. Now open the `rssrvpolicy.config` file, also located in the `ReportServer` folder. Here you'll add code group for the custom assembly to grants FullTrust permission for your extension.

5. Add a new CodeGroup section to the `rssvpolicy.config` file:

```
<CodeGroup class="UnionCodeGroup" version="1"
          PermissionSetName="FullTrust"
          Name="FSIExtensionCodeGroup"
          Description="Code group for data extension">
          <IMembershipCondition
               class="UrlMembershipCondition" version="1"
               Url="C:\Program Files\Microsoft SQL Server\
                    MSSQL\Reporting Services\
                    ReportServer\bin\FSIDataExtension.dll" />
</CodeGroup>
```

   Save the changes to the `rssvpolicy.config` file.

If the data processing extension was deployed successfully, it will be included in the list of available data sources.

Deploying a Data Processing Extension to Report Designer

Report Designer uses data processing extensions at design time to retrieve and display data. Therefore, you should also deploy your custom extension to Report Designer. The process is similar: you copy the same file(s) to a folder under Visual Studio's IDE folder and edit a different configuration file. The steps for deploying a data processing extension to Report Designer are as follows:

1. Copy the DLL file(s) generated by the compiler to the Report Designer's directory. The default location of this directory is `C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies`.

2. Open the `RSReportDesigner.config` file (which is located in the same directory) and add an entry in the configuration file for your data processing extension. Locate the `Data` element in the configuration file and insert the following:

```
<Extensions>
  <Data>
    <Extension Name="FSIDataExtension"
               Type="Microsoft.FSIDataExtension.FSIConnection,
               FSIDataExtension" />
  </Data>
</Extensions>
```

   The `Name` attribute must be unique in the configuration file.

3. Finally, you must edit the `rspreviewpolicy.config` file, also located in the same folder and add code group for the custom assembly to grant FullTrust permission for your

extension. The configuration file is located by default in the `C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies` folder; your entry should looks like this:

```
<CodeGroup class="UnionCodeGroup" version="1"
      PermissionSetName="FullTrust"
      Name="FSIDataExtension"
      Description="Item's description">
<IMembershipCondition class="UrlMembershipCondition"
       version="1"
       Url="C:\Program Files\Microsoft Visual Studio 8\
       Common7\IDE\PrivateAssemblies\FSIDataExtension.dll" />
</CodeGroup>
```

The Bottom Line

Explore the extension types. Reporting Services is based on an architecture that allows you to extend its functionality by implementing custom extensions.

Master It How many types of extensions are there?

Build security extensions. Security extensions are implemented as classes that expose specific functionality. This functionality is predetermined, and it's provided to you by means of interfaces: an interface is a class that contains property and method definitions, but no code. It's your responsibility to implement the members of an interface. Once an interface is implemented, you can incorporate it into Reporting Services.

Master It Which are the interfaces you must implement for a security extension?

Deploy security extensions. Security extensions are compiled in DLL files, which you must copy into specific folders of the Reporting Services installation. In addition, you must modify certain configuration files.

Master It Where would you copy the assembly of a security extension?

Build data processing extensions. Data processing extensions are implemented as classes that expose specific functionality. This functionality is predetermined, and it's provided to you by means of interfaces: an interface is a class that contains property and method definitions, but no code. It's your responsibility to implement the members of an interface. Once an interface is implemented, you can incorporate it into Reporting Services.

Master It Which are the interfaces for a data processing extension?

Deploy data processing extensions. Data processing extensions are compiled in DLL files, which you must copy into specific folders of the Reporting Services installation. In addition, you must modify certain configuration files.

Master It Which configuration files should you edit in the process of deploying a data processing extension?

Chapter 13

Report Definition Language

Now it's time to take a closer look at the Report Definition Language (RDL) files generated by the visual tools of the Business Intelligence Development Studio. All the information regarding the data itself and the appearance of the data on the report is stored in RDL files. These files contain all the information needed to render a report, except for the data, which is read from the data source when the report is executed. RDL is a language that describes the layout of the report, just as HTML is a text file that describes a web page with rich content. RDL is a markup language, like HTML and XML, and it lacks the structures of programming languages such as conditionals and loops.

For the purposes of this chapter I've created a simple report to print the names and addresses of the Northwind sample database. If you haven't installed the Northwind database, you'll have to use a different query to retrieve data from another database. We're not really interested in the appearance of the report, or the exact data we'll present; we only care about the structure of the information stored in the report definition file.

Microsoft could have used any proprietary format for storing the report definition, but developers wouldn't be able to extend the architecture of SQL Server Reporting Services. A better method is needed for an open architecture, and this method is an XML file. The RDL files are indeed XML files with a specific structure. The XML document that describes a report has a specific structure, which is called a *schema*, and you can download it from Microsoft. The schema describes the structure of the data, and a report definition file is a different implementation of this schema. In other words, all report definition files have the same structure, but they describe different data.

In this chapter you will learn how to:

- Understand the report definition schema
- Define a report's data source and data sets
- Create report and query parameters
- Use the `ReportItems` element
- Use the `XmlTextWriter` class

What's in an RDL File

Let's start our exploration of RDL by looking at a file generated by the wizard. Create a new report, or open a report project from the previous chapters. Then right-click the RDL file in the Solution Explorer and select View Code from the context menu. You will see a long XML file in the editor, as shown in Figure 13.1. This file contains all the information necessary for the Report Server to render the report. You can modify the report with visual tools and view the changes in the XML file, or edit the XML file and see the changes on the design surface.

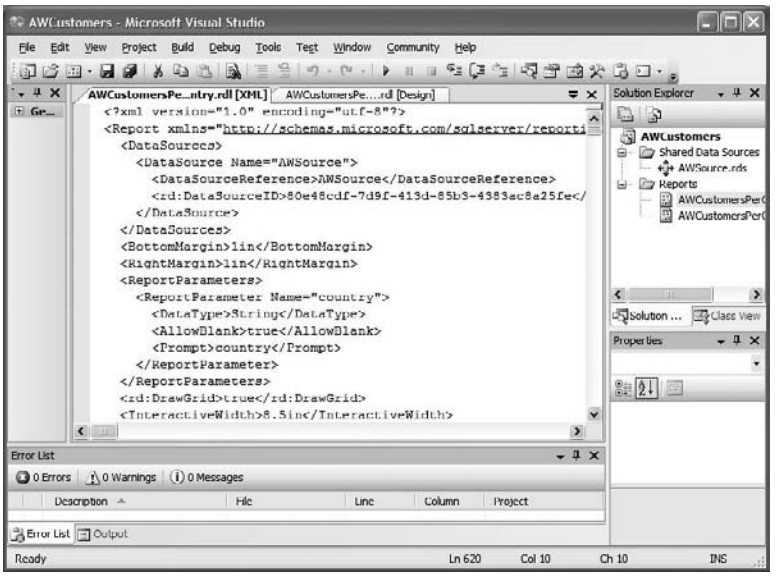


FIGURE 13.1
Viewing the RDL description of a report

Inevitably, there will be a `Style` element in the document (there are many such elements). Locate a `Style` element, which is delimited by the `<Style>` and `</Style>` tags. This element determines the style of a report item (a header, a cell in the report's details element, and so on). Note the parent element, so that you'll know where to look for the changes. In the `Style` element you'll find a `Color` element that looks like this:

```
<Color>SteelBlue</Color>
```

As you can guess, this segment of the document specifies the color of a specific item. Change the color name to `Red` and then switch to the report's Layout or Preview mode. The IDE will take a moment to redraw the design surface, but when it's done you will see that the corresponding report item has changed color from `SteelBlue` to `Red`.

Near the end of the RDL file you will find the query that retrieves the desired data from the server. Find the `CommandText` element under the `Query` element; look for something like this:

```
<Query>
  <CommandText>SELECT ... </CommandText>
  ...
</Query>
```

If you don't want to edit the `DataSource` interactively, you can change the query in the report definition file. If you add or remove fields, you'll have to edit the RDL file elsewhere to display the additional columns. All the specifications you've created for the report with visual tools are stored in this document. As you edit the report by adding more columns, changing their widths, grouping the report's rows, and so on, the RDL file is being continuously modified by the designer.

In this chapter, you'll learn how to generate new reports using the Report Definition Language. The first step is to understand the schema that describes the reports.

The Structure of an RDL File

An RDL file is made up of sections, which are delimited by tags such as <PageHeader> and </PageHeader> or <Body> and </Body>. In XML, the tags that describe the sections of the document are called *elements*. In this chapter I'll use the term *element*, but for all practical purposes you can consider them as section separators; as you will see, an RDL file consists of nested elements. The Body element contains a ReportItems element, which in turn contains multiple Table elements. Each Table element, in turn, has a Header element and a Details element, which are also made up of finer elements. We'll explore the structure of an RDL file in detail in this section. Once you have a basic understanding of the structure of a report, you'll be able to generate reports from within your code.

The top element of an RDL file is the Report element, which contains all other elements. The entire report definition is contained between the <Report> and </Report> tags. Within the Report element of the document you can insert any of the elements shown in Table 13.1, which correspond to the basic sections of the report.

TABLE 13.1: Elements within the *Report* Element

ELEMENT	PURPOSE
<PageHeader>	The header of each page
<PageFooter>	The footer of each page
<Body>	The report's body
<EmbeddedImages>	The report's embedded images
<DataSources>	The report's data sources
<DataSets>	The report's data sets
<Parameters>	The report's parameters

Many of these elements are made up of ReportItems, which are text boxes, lines, rectangles, images, and so on; they're the items you can drop on the report's layout from the Toolbox. The most common is the Textbox item, where all fields are displayed. There are also three ReportItems that determine the overall appearance of the report, as shown in Table 13.2.

TABLE 13.2: The *ReportItem* Elements that determine the structure of the report

ELEMENT	PURPOSE
<Table>	For tabular reports
<Matrix>	For matrix reports
<Chart>	For graphs

Here are two `ReportItems` for displaying the State and Country fields:

```
<ReportItems>
    <Textbox Name="State">
    <Style>
      ...
    </Style>
    <Value>=Fields!State.Value</Value>
  </Textbox>
  <Textbox Name="Country">
    <Style>
      ...
    </Style>
    <Value>=Fields!Country.Value</Value>
  </Textbox>
</ReportItems>
```

To insert field values, RDL uses the `Fields` keyword, followed by an exclamation mark and the name of a field from the current data set. The expression `Fields!State` represents the State field, which among other properties provides the `Value` property, which returns the value to be displayed on the report. The `Style` element determines the appearance of the corresponding item; you can specify many properties, such as the font, background color, border of the text box, and so on. I've used an ellipsis (. . .) to indicate that there are more elements in each section, which were omitted for clarity. Notice the expressions that set the values of the text boxes: they're identical to the ones generated by the wizard when you drop a field onto the report's design surface.

The main section of the report is contained in the `Body` element, which has the following structure:

```
<Body>
  <ReportItems>
    <Table>
      <TableColumns>
        <TableColumn> ... </TableColumn>
        <TableColumn>  ... </TableColumn>
      </TableColumns>
      <Header>
        <TableRows>
          <TableRow>
            <TableCells>
              <TableCell>
                <ReportItems> ... </ReportItems>
              </TableCell>
              <TableCell>
                <ReportItems> ... </ReportItems>
              </TableCell>
            </TableCells>
          </TableRow>
        </TableRows>
```

```
      </Header>
      <Details>
        <TableRows>
          <TableRow>
            <TableCells>
              <TableCell>
                <ReportItems>  ... </ReportItems>
              </TableCell>
              <TableCell>
                <ReportItems>  ... </ReportItems>
              </TableCell>
            </TableCells>
          </TableRow>
        </TableRows>
        </Details>
    </Table>
  </ReportItems>
</Body>
```

The `Body` element contains a `ReportItems` element, where the body of the report is defined. For simple reports there's only one report item in the report's body; it specifies the type of the report and it can be a `Table` item, a `Matrix` item, or a `Chart` item. Reports with subreports may contain multiple report items in their `Body` element. Now, depending on type of the report, you must specify the following additional information: for a `Table` item, you must specify the columns of the table as `TableColumn` subelements under the `TableColumns` element. For a `Matrix` item, you must specify the columns of the matrix as `MatrixColumn` subelements under the `MatrixColumns` elements. Once the columns have been specified, you must insert the `Header` and `Details` elements. Both items contain a `TableRows` element, which is a collection of `TableRow` elements. The `Header` element usually contains a single row, but reports with multiple header rows are not uncommon. Finally, the `TableRow` element contains a `TableCells` element, which is made up of `TableCell` elements, one for each cell in the row.

The `ReportItem` element is the basic entity for displaying data. The most common report item is the `Textbox` item. Other report items are `Rectangle`, `Image`, and `SubReport`. There's a correspondence between the `TableCell` elements of the `Header` and those of the `Details` element. The difference is that the `Header` elements contain the column captions, while the `Details` elements contain the fields to be displayed on the report body. The following statements set up two header columns:

```
<TableCells>
  <TableCell>
    <ReportItems>
      <Textbox Name ="hdrCompany">
        <Value>Company Name</Value>
      </Textbox>
    </ReportItems>
  </TableCell>
  <TableCell>
    <ReportItems>
```

```
        <Textbox Name ="hdrAddress">
          <Value>Customer Address</Value>
        </Textbox>
      </ReportItems>
    </TableCell>
  </TableCells>
```

The equivalent segment in the `Details` section should contain the following tags:

```
<TableCells>
  <TableCell>
    <ReportItems>
      <Textbox Name = "dtlCompany">
        <Value>=Fields!Company1.value</Value>
      </Textbox>
    </ReportItems>
  </TableCell>
  <TableCell>
    <ReportItems>
      <Textbox Name = "dtlAddress">
        <Value>=Fields!Address1.value</Value>
      </Textbox>
    </ReportItems>
  </TableCell>
</TableCells>
```

The `Textbox` element has a `Value` subelement, which is set to a string for the `Header` and to an expression for the `Details` element. The expression is the name of the field we want to display in the text box.

Creating a Report Definition File

In this section you're going to build a report manually. It will give you a good understanding of the structure of the RDL file. This information will come in handy later, when you'll write applications to automate the creation of report files. If you don't want to follow the steps, you can download the ReportDefinition project from this book's page at the publisher's site. If you don't have Visual Studio 2005 installed on your computer, you can download the trial version of the product from `http://msdn.microsoft.com/VB`.

Start Visual Studio, create a new Windows project, and add a new XML file to the project. To do so right-click the project's name in Solution Explorer and select Add ➢ New Item from the context menu. When the Add New Item dialog box appears, select the XML File template, specify a name for the file, and click OK. The name I've used for this file in the ReportDefinition project is `Report.xml`. When the new item is added to the Solution Explorer, it opens in the IDE and you will see a single statement:

```
<?xml version="1.0" encoding="utf-8" ?>
```

If you look at the XML file's Properties window, you will see that the `Schemas` property is blank. The XML file doesn't conform to any schema, and you can edit it to your heart's content. The editor will make sure that the file you create is well formed, but that's all. It has no way to enforce

the generation of an XML document that follows the Report Definition schema. Let's locate the appropriate schema and use it in our code. Start your browser and navigate to the following URL:

```
http://schemas.microsoft.com/sqlserver/reporting/2005/01/
reportdefinition/ReportDefinition.xsd
```

This URL may change in the future, but you should always be able to locate the `ReportDefinition.xsd` file at `http://msdn.microsoft.com`. The XSD file is also an XML file that describes the schema all reports must adhere to. Save the document on your computer's filesystem as `ReportDefinition.xsd` by selecting File ➢ Save As in Internet Explorer. Use a folder that doesn't belong to a project, because you may use this file with many projects. The report definition schema will simplify the manual editing of RDL files enormously, as you will see shortly. Click the ellipsis button next to the `Schemas` property in the Properties window to open the XSD Schemas dialog box, shown in Figure 13.2. Here you can select any schema and apply it to the open XML document. Locate the `ReportDefinition.xsd` file you just downloaded and then click OK to close the dialog box and apply the XSD file to the XML document.

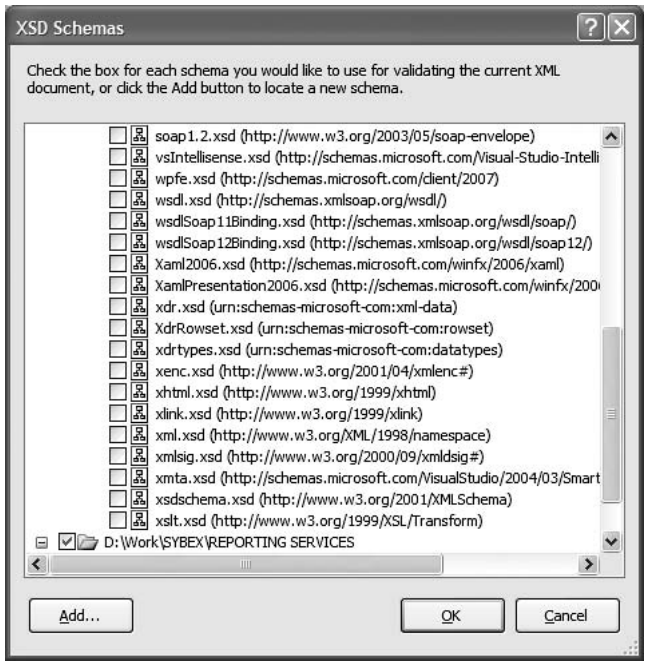


FIGURE 13.2
Applying a schema to an XML document

If you open the `ReportDefinition.xsd` file, you will see that it's an XML file, not unlike the ones generated by the wizard. The difference is that it contains no data, just data definitions. Switch back to the XML editor in Visual Studio and start editing it.

NOTE

XML is case sensitive, something that VB developers are not accustomed to. The most common source of errors in XML files is the casing of tag names, or inconsistent casing of user-supplied strings in the document.

Enter an opening angle bracket in the next line, and a drop-down list opens containing the keywords that may appear at the current location. Select `Report` and press Enter. The editor inserts the `xmlns` attribute, and you can type the closing bracket. The editor inserts the closing tag, `</Report>`, and leaves the cursor between the two tags. This is where you must specify all properties of the report. All elements must be nested between the `<Report>` and `</Report>` tags, since the `Report` element is the root element for the document.

The *DataSources* Element

Type an opening angle bracket again, and this time you'll see a list with more items. These are the items you can place inside a report: `Body`, `PageFooter`, `PageHeader`, `DataSources`, and `DataSets`. The main element that contains the entire report is the `Body` element, but we usually start with the `DataSources` element. Enter the opening bracket as usual and from the list select `DataSources`. The editor will insert the closing tag for you and will place the cursor between the two new tags. As you can see, creating an XML document that's based on a specific schema is straightforward, as long as you have some understanding of the structure of the document you're building. To familiarize yourself with the report definition schema, create a sample report and use the wizard-generated RDL file as a guide.

The `DataSources` element contains one or more `DataSource` elements. Insert a `<DataSource>` tag and the editor displays another list, this time containing the attributes of the `DataSource` element. The `DataSource` element must have a `Name` attribute. Select `Name` in the list and the editor inserts the equals sign and a pair of double quotes. You must type the name of the current data source between the quotes. Enter a string like `"CustomersSource"` and then the closing angle bracket. The closing bracket must appear after the closing quotation mark, of course. Insert a new angle bracket and you will see a list of the elements that may appear within a `<DataSource>` tag: `ConnectionProperties`, `DataSourceReference`, and Transaction. Select `ConnectionProperties` and type the closing bracket. After the `<ConnectionProperties>` tag, insert the `<ConnectionString>` tag, selecting it from the list as usual. Between the `<ConnectionString>` tag and its closing tag, you must insert the connection string that the Report Server will use to connect to the database. Enter the following string between them:

```
data source=localhost;integrated security=True;
initial catalog=Northwind
```

There's no need to enter any quotes, just the string; also, it's OK to break lines arbitrarily in an XML document.

The *DataSets* Element

So far you've created the data source: you've specified where the data will come from. The next step is to specify the actual data. Move the pointer after the `<>tag` and press Enter. Insert the tags `<DataSets>` and `<DataSet>`. The `DataSet` element must have a name, which you specify as an attribute of the `<DataSet>` tag. Then insert the `<Query>` tag. The `Query` element should include the following elements: `<DataSourceName>` (this one should match an existing data source in the document), `<CommandType>`, and `<CommandText>`. Here's a complete `Query` element:

```
<Query>
  <DataSourceName>DSource</DataSourceName>
  <CommandType>Text</CommandType>
  <CommandText>SELECT CompanyName AS Company,
      Address AS Address FROM Customers</CommandText>
</Query>
```

After the `Query` element you must enter the `Fields` element, and then a `Field` element for each table column under it. Each `Field` element must have a name, which is a name of your choice. This is how you will identify the field in the XML document. Within the `Field` element insert the `DataField` element, which is the field's name in the query. Use a different name only if the query contains identical field names from multiple tables, or if the names returned by the query are cryptic. Even then, it's recommended that you use the `AS` clause in your query to return the desired field names. Here's a complete `Fields` element for our example:

```
<Fields>
  <Field Name="Company">
    <DataField>Company</DataField>
  </Field>
  <Field Name="Address">
    <DataField>StreetAddress</DataField>
  </Field>
  <Field Name="City">
    <DataField>City</DataField>
  </Field>
  <Field Name="Country">
    <DataField>Country</DataField>
  </Field>
</Fields>
```

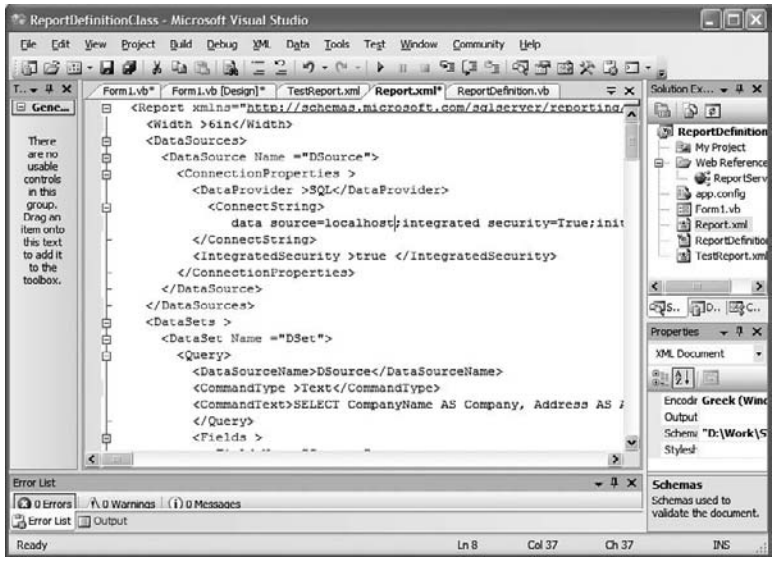


FIGURE 13.3
Inserting the `DataSources` and `DataSets` elements in an XML file that describes a new report

Note that the database field StreetAddress is mapped to the `Address` field for the report, and from now on you'll use the Address name for the `StreetAddress` field. So far you've specified the data that will be used in your report as well as the source of the data. This corresponds to the first few steps of the Data Source wizard, where you specify the connection string and

the query. Now press Enter after the </DataSets> tag and insert the <Body>tag. The actual report definition will be inserted between these <Body> and </Body> tags. As soon as you enter an opening angle bracket in the Body element, you see that only a few elements may appear in this section of the document: Columns, ColumnSpacing, Height, ReportItems, and Style. The Height element, which specifies the column's width, is mandatory, so enter the following item:

```
<Height>6.5in</Height>
```

and then enter the <ReportItems> tag. Under the <ReportItems> tag, you can insert either a <Table> or a <Matrix> tag, depending on the type of report. For this example, enter the <Table> tag, and then type another opening angle bracket. This time you will see a long list of elements you may nest in the Table element. One of them is the DataSetName element, which specifies the name of the DataSet (it should be the same as the one you specified earlier in the document).

The *Body* Element

In the Body element you must insert a TableColumns element. Use the <TableColumns> tag to create a section with the report's columns. This section is made up of TableColumn elements, and you must use a <TableColumn> tag for each column in the report. Each column has a name, which is specified as attribute of the <TableColumn> tag, and a width, which is specified with the <Width> tag. Here's a list of column definitions for the sample report:

```
<TableColumns>
  <TableColumn>
    <Width>2in</Width>
  </TableColumn>
  <TableColumn>
    <Width>2.5in</Width>
  <TableColumn>
    <Width>1.5in</Width>
  </TableColumn>
  <TableColumn>
    <Width>1in</Width>
  </TableColumn>
</TableColumns>
```

The *Header* Element

Then comes the Header element, where you specify the headers of the report. This element contains a TableRows element, which is made up of TableRow elements. Each row has a height, which is specified with the Height element, and a TableCells element, which is made up of TableCell elements. Each TableCell element, in turn, contains a ReportItems element. The ReportItems element, however, is not made up of ReportItem elements — in fact, there's no ReportItem element. The ReportItems element may contain one of several items like rectangles, lines, tables, and text boxes.

The Textbox Item

The most common item in a report is the `Textbox` element. To display the three fields we specified earlier in the `Fields` section of the report, insert the following statements in the `Header` element of the XML file:

```
<Header>
  <TableRows>
    <TableRow>
      <Height>0.25in</Height>
      <TableCells>
        <TableCell>
         <ReportItems>
            <Textbox Name ="hdrCompany">
              <Value>CompanyName</Value>
            </Textbox>
          </ReportItems>
        </TableCell>
        <TableCell>
         <ReportItems>
            <Textbox Name ="hdrAddress">
              <Value>Customer Address</Value>
            </Textbox>
          </ReportItems>
        </TableCell>
      <TableCell>
          <ReportItems>
            <Textbox Name ="hdrCity">
             <Value>City</Value>
           </Textbox>
          </ReportItems>
        </TableCell>
      <TableCell>
          <ReportItems>
            <Textbox Name ="hdrCountry">
             <Value>Country</Value>
           </Textbox>
          </ReportItems>
        </TableCell>
      </TableCells>
    </TableRow>
  </TableRows>
</Header>
```

All data in a report are displayed in `Textbox` items, which must have a name and a value. Each `Textbox` item's value in the report's `Header` section is the column's caption, which is usually a string.

The *Style* Element

You can control the appearance of the `Textbox` item with the `Style` element. As soon as you enter the `<Style>` tag within a `Textbox` element, you will see that it supports a large number of properties, such as the `BackgroundColor`, `BorderColor`, `Color`, `Font`, `TextAlign`, and many more. You can set all the properties you would normally set in the Properties window of a Textbox control in the Layout Designer. The colors can be specified by name, which means that you will most likely locate them in the designer, write down their names, and reuse them in your code. Here's a sample `Style` element:

```
<Style>
  <PaddingLeft>2pt</PaddingLeft>
  <PaddingBottom>2pt</PaddingBottom>
  <PaddingRight>2pt</PaddingRight>
  <PaddingTop>2pt</PaddingTop>
  <BackgroundColor>WhiteSmoke</BackgroundColor>
  <Color>DarkGray</Color>
</Style>
```

There are other elements you can use to adjust the appearance of individual `Textbox` items, and you can experiment with them interactively. Design a text box interactively in the designer and then see the RDL file generated by the editor.

The *Details* Element

After the `Header` element comes the `Details` element, which represents the main area of the report (the grid with the data). The `Details` element has the same structure as the `Header` element, but the value of the `Textbox` items is set to the name of the field we want to display on the report. Here's the definition of the `Textbox` item where the company name is displayed:

```
<Textbox Name="txtCompany">
  <Value>=Fields!CompanyName.Value</Value>
</Textbox>
```

The `<Value>` tag is mandatory, even if it's empty. If an element does not have a value, you can open and close it with a single tag: `<Value/>`. The `Name` attribute is also mandatory, even though it's not used anywhere else in the document.

Add three more `Textbox` items on the report to display the address fields:

```
<Textbox Name="txtAddress">
  <Value>=Fields!StreetAddress.Value</Value>
</Textbox>
<Textbox Name="txtCity">
  <Value>=Fields!City.Value</Value>
</Textbox>
<Textbox Name="txtCountry">
  <Value>=Fields!Country.Value</Value>
</Textbox>
```

Deploying the New Report

You have created a new report entirely in code. The RDL file represents a new report, which you can publish on the Report Server so that users will see it along with the other reports. Place a new button on the main form of the application (it's the Read Report Definition and Publish Report button of the ReportDefinition sample project). In this button's `Click` event handler, you must insert the code to read the report's definition and create a new report with the `CreateReport` method of the ReportService2005 web service, as explained in Chapter 11.

As you recall from Chapter 11, we'll use the ReportingService2005 web service to publish a new report. This web service exposes the `CreateReport` method, which accepts as an argument the definition of the new report (the RDL file we just created). The method doesn't accept a filename as an argument but an array of bytes with the contents of the file. It also accepts a few more arguments, such as the report's path on the Report Server and the name of the new report.

To use the Report Server's web service, add the appropriate web reference to your project. Right-click the project name and from the context menu select Add Web Reference. When the Add Web Reference dialog box appears, click the Web Services on the Local Machine link to see the web services exposed by your computer. On the next screen select Report Services 2005 Web service. Set the name of the web service to **ReportServer** (this is how you will reference the web service from within your project) and click Add Reference to dismiss the dialog box. We have a reference to the Report Services 2005 component and can now implement the `PublishReport` subroutine, shown here:

```
Private Sub PublishReport()
    Dim reportPath As String = "/"
    Dim strm As New StreamReader("Report.xml")
    Dim definition As String = strm.ReadToEnd()
    Dim RS As New ReportServer.ReportingService2005
    RS.Credentials = _
            System.Net.CredentialCache.DefaultCredentials
    Try
        Debug.WriteLine(definition)
        RS.CreateReport("AUTOGENERATEDREPORT", ➢
                reportPath, True, _
                System.Text.Encoding.UTF8.GetBytes(definition), ➢
                Nothing)
    Catch ex As Exception
        MsgBox("Failed to publish report!" & vbCrLf & ex.Message)
        Exit Sub
    End Try
    MsgBox("Report deployed successfully. " & ➢
           "Direct your browser  to the Report Server's " & ➢
           "main page to see the AUTOGENERATEDREPORT report")
End Sub
```

Now you can start your browser, navigate to the Report Server, and view the report you just created. Editing the report definition file manually is a straightforward process thanks to the schema we have applied to the XML document. The editor validates the XML document against its schema as you edit it and underlines the tags in error. If the XML file is error free, you have created a valid report definition. It may look different than what you had in mind, but it's

a valid report. You can always go back to the editor and modify it until you get it right. The only way to test a report without the visual tools of Reporting Services is to publish it and then open it with your browser.

Specifying Report Parameters

If the report contains parameters, you must insert them in a `ReportParameters` element. The `ReportParameters` element contains a number of `ReportParameter` elements, one per report parameter. Each `ReportParameter` element must have a name and a data type, as well as a value, if you want to specify it from within your code. Here's a typical parameter definition:

```
<ReportParameters>
  <ReportParameter Name="country">
    <DataType>String</DataType>
    <AllowBlank>true</AllowBlank>
    <Prompt>Enter a country name</Prompt>
  </ReportParameter>
</ReportParameters>
```

You can also set the parameter's value with the `DefaultValue` element, and if you want to hide the prompt for the parameter, insert a `<Hidden>` tag and set it to False. To specify a parameter's value in the RDL file, use the `Values` element, which is made up of `Value` elements, as shown here:

```
<DefaultValue>
  <Values>
  <Value>United States</Value>
  </Values>
</DefaultValue>
```

Of course, it rarely makes sense to hardcode parameter values, but an application that creates reports on the fly could pick values from other controls on the form and insert them in the report definition file. Let's say you have a form that contains the `txtCountry` TextBox control and you want to pass the value of the control as a report parameter. The following statements emit the preceding XML segment using the control's value as the `country` parameter's value:

```
Writer.WriteLine("<ReportParameters>")
Writer.WriteLine("<ReportParameter Name=""country""">"
Writer.WriteLine("<Value>" & txtCountry.Text.Trim & "</Value>")
```

The *`Writer`* variable represents a properly initialized `StreamWriter` object that points to a file or memory stream that will accept the RDL code generated by your code.

If it's a multivalued parameter, insert additional `<Value>` tags for all possible values. In addition, you must set the `MultiValue` element to True with the following tag:

```
<MultiValue>true</MultiValue>
```

If the parameter's valid values are the result of a query, a common situation in designing reports, you must insert a `ValidValues` element instead of the `DefaultValue` element. In the `ValidValues` element insert a reference to the `DataSet` that returns the valid values (country

names, for instance) and specify the field to be displayed with the <ValueField> tag, as shown in the following sample code:

```
<ValidValues>
  <DataSetReference>
    <DataSetName>DSCountryNames</DataSetName>
    <ValueField>Country</ValueField>
    <LabelField>Select a country name</LabelField>
  </DataSetReference>
</ValidValues>
```

Creating Reports in Code

By now you have a basic understanding of the structure of a report and how the Report Definition Language is used to store the description of a report in an RDL file. Generating reports manually in RDL is not practical and offers no advantages over the visual tools. In this section, you'll learn how to create reports in RDL from within a Windows application and then publish the reports to the Report Server.

The *XmlTextWriter* Class

The .NET Framework provides the XmlTextWriter class, which simplifies the task of writing structured XML documents, such as report definition files. The few methods of the XmlTextWriter class that are of interest for the purposes of creating XML documents are the following:

WriteStartElement(element) The WriteStartElement method writes the opening tag for the element you pass to the method as a string argument.

WriteEndElement The WriteEndElement method inserts the closing tag for the most recently opened tag. This method does not accept an argument; it automatically figures out the tag that can be closed.

WriteAttributeString(attribute, value) The WriteAttributeString method writes an attribute and its value as specified by the two arguments in the most recently opened element.

WriteElementString(element, value) The WriteElementString method writes the opening and closing tags of the specified element and the value specified by the second argument. The following statements:

```
writer.WriteStartElement("DataSources")
writer.WriteStartElement("DataSource")
writer.WriteAttributeString("Name", "DSource")
writer.WriteEndElement
writer.WriteEndElement
```

will produce the following output:

```
<DataSources>
  <DataSource Name = "DSource">
  </DataSource>
</DataSources>
```

The *writer* variable is an instance of the XmlTextWriter class, which must be declared with the following statement:

```
Dim writer As New System.Xml.XmlTextWriter ➢
          (strm, System.Text.Encoding.UTF8)
```

To use the XmlTextWriter class in your project you must import the System.Xml namespace to your project.

Here's an example of the WriteElementString method, which inserts an element and its value. The following statements create a Query element, which contains the definition of a data source and the query to be executed against it:

```
writer.WriteStartElement("Query")
writer.WriteElementString("DataSourceName", "DSource")
writer.WriteElementString("CommandType", "Text")
writer.WriteElementString("CommandText", _
                          "SELECT * FROM Customers")
writer.WriteEndElement()
```

The actual output produced by these statements is shown next:

```
<Query>
  <DataSourceName>DSource</DataSourceName>
  <CommandType>Text</CommandType>
  <CommandText>SELECT * FROM Customers</CommandText>
</Query>
```

With the four methods of the XmlTextWriter class, you can build an XML document step by step, as long as you know the hierarchy of its elements. Our recommendation is to start with an existing RDL file that contains sections similar to ones you're interested in, and once you know the elements, attributes, and values you want to insert into the RDL file, use the XmlTextWriter method to create the file.

Building a Report Definition in VB

Let's build the sample report of the preceding section from within a VB application. The code discussed in this section is included in the ReportDefinition sample project. Open Visual Studio and create a new Windows application with Visual Basic. Visual Studio will add the Form1 form to the project. Double-click the form to open the code editor.

Start by importing the following namespaces in your form (insert the statements shown next at the top of the editor's window):

```
Imports System.Xml.Serialization
Imports System.IO
Imports System.Data.SqlClient
Imports System.Xml.XmlTextWriter
```

Then place a button on the form, the Create Report Definition and Publish Report button of the ReportDefinition sample project. In the button's Click event handler, call two procedures for creating and publishing the report:

```
CreateReport()
PublishReport()
```

Now we can flesh out these two procedures, starting with the `CreateReport` procedure. First we must create a file with the report's definition. Create a `FileStream` object that represents the file where the XML document will be stored with the following statements:

```
Dim strm As FileStream
strm = File.OpenWrite("C:\AutoGeneratedReport.rdl")
strm.SetLength(0)
```

The new file will be created in the root of the C: drive, so that you will find it easily. The last statement truncates the file, should it exist. We'll use the `AutoGeneratedReport.rdl` file to publish the report on Report Server and then delete it.

Next create an instance of the `XmlTextWriter` object, which you'll use to write the necessary element to the output file:

```
Dim writer As New System.Xml.XmlTextWriter _
        (strm, System.Text.Encoding.UTF8)
```

To produce a readable file, force the `XmlTextWriter` object to indent the XML document properly:

```
writer.Formatting = System.Xml.Formatting.Indented
```

Now you're ready to start writing elements to the XML document. Start by inserting the `<Report>` tag, which requires the `xmlns` attribute:

```
writer.WriteStartElement("Report")
writer.WriteAttributeString("xmlns", Nothing, ➢
           "http://schemas.microsoft.com/sqlserver/ ➢
            reporting/2003/10/reportdefinition")
```

Then write the `Width` element with the appropriate value:

```
writer.WriteElementString("Width", "6in")
```

We're done with the file's header and we're ready to populate the various sections of the report, starting with the `DataSources` element. Here we'll specify where the data will come from (basically, we'll tell the report engine how to connect to the appropriate database). The following statements will insert the `DataSources` element into the XML document:

```
writer.WriteStartElement("DataSources")
writer.WriteStartElement("DataSource")
writer.WriteAttributeString("Name", "DSource")
writer.WriteStartElement("ConnectionProperties")
writer.WriteElementString("DataProvider", "SQL")
writer.WriteElementString("ConnectString", CNString)
writer.WriteElementString("IntegratedSecurity", "true")
writer.WriteEndElement()
writer.WriteEndElement()
writer.WriteEndElement()
```

The *CNString* variable should be initialized as follows:

```
Dim CNString As String = "data source=localhost;" & ➢
     "initial catalog=Northwind;integrated security=True"
```

Then come the `DataSets`, `Query`, and `Fields` sections of the report. These sections are quite similar to the examples shown already, so I will not list the code. You can always open the `ReportDefinition` project and examine its code.

Inserting the Report's Body Element

Let's jump to the `Body` section of the report, which is the core of the report (the data to be displayed on the grid). The following code segment generates an XML segment identical to the one you created manually in the preceding section. The `Body` element contains a `ReportItems` element, which in turn contains the `Table` element. In the `Table` element, the code inserts the definition of the report's columns and then each page's header.

```
writer.WriteStartElement("Body")
writer.WriteElementString("Height", "6.5in")

' ReportItems element
writer.WriteStartElement("ReportItems")
' Table element
writer.WriteStartElement("Table")
writer.WriteAttributeString("Name", Nothing, "tbl1")
writer.WriteElementString("DataSetName", "DSet")
' Table Columns
writer.WriteStartElement("TableColumns")
writer.WriteStartElement("TableColumn")
writer.WriteElementString("Width", "2.5in")
writer.WriteEndElement() ' TableColumn

writer.WriteStartElement("TableColumn")
writer.WriteElementString("Width", "3.5in")
writer.WriteEndElement() ' TableColumn
writer.WriteEndElement() ' TableColumns
' Header Row
writer.WriteStartElement("Header")
writer.WriteStartElement("TableRows")
writer.WriteStartElement("TableRow")
writer.WriteElementString("Height", ".25in")
writer.WriteStartElement("TableCells")
writer.WriteStartElement("TableCell")
writer.WriteStartElement("ReportItems")
' Textbox for 1st column
writer.WriteStartElement("Textbox")
writer.WriteAttributeString("Name", Nothing, "txtCompanyName")
writer.WriteElementString("Value", "Company Name")
writer.WriteEndElement() ' Textbox
writer.WriteEndElement() ' ReportItems
writer.WriteEndElement() ' TableCell
```

```
' Textbox for 2nd column
writer.WriteStartElement("TableCell")
writer.WriteStartElement("ReportItems")
writer.WriteStartElement("Textbox")
writer.WriteAttributeString("Name", Nothing, "txtAddress")
writer.WriteElementString("Value", "Customer Address")
writer.WriteEndElement() ' Textbox
writer.WriteEndElement() ' ReportItems
writer.WriteEndElement() ' TableCell
writer.WriteEndElement() ' TableCells
writer.WriteEndElement() ' TableRow
writer.WriteEndElement() ' TableRows
writer.WriteEndElement() ' Header
```

Inserting the Report's Details Element

The core of the report is the `Details` element, which is also part of the `Table` element under the `Body` element. The `Details` element contains a `TableRows` element, which in turn contains a `TableRow` element for each row of data. Finally, each `TableRow` has its own `TableCells` element, which contains a `TableCell` element for each cell. To specify the data to be displayed in each cell, the code inserts a `Textbox` item and sets its `Value` property to the name of the field to be displayed on the corresponding item.

```
writer.WriteStartElement("Details")
writer.WriteStartElement("TableRows")
writer.WriteStartElement("TableRow")
writer.WriteElementString("Height", ".25in")
writer.WriteStartElement("TableCells")
writer.WriteStartElement("TableCell")
writer.WriteStartElement("ReportItems")
' First cell in row contais the Company field
writer.WriteStartElement("Textbox")
writer.WriteAttributeString("Name", "TXTCOMPANY")
writer.WriteElementString("Value", "=Fields!Company.Value")
writer.WriteEndElement() ' TextBox
writer.WriteEndElement() ' ReportITems
writer.WriteEndElement() ' TableCell
' Second cell in row contains the Address field
writer.WriteStartElement("TableCell")
writer.WriteStartElement("ReportItems")
writer.WriteStartElement("Textbox")
writer.WriteAttributeString("Name", Nothing, "TXTADDRESS")
writer.WriteElementString("Value", "=Fields!Address.Value")
writer.WriteEndElement()  ' TextBox
writer.WriteEndElement()  ' ReportITems
writer.WriteEndElement()  ' TableCell
writer.WriteEndElement()  ' TableCells
writer.WriteEndElement()  ' TableRow
writer.WriteEndElement()  ' TableRows
writer.WriteEndElement()  ' Details
```

NOTE

Because you can't indent VB code lines as you wish, you must somehow keep track of the tags closed with the `WriteEndElement` method. The only way to keep track of the closing tags is to append a comment with the name of the tag being closed by each call to the method. Otherwise, you'll find it hard to match the opening and closing tags in your code, especially since you can't indent code lines in the editor of Visual Studio.

Here's the output generated by the preceding statements:

```
<Details>
  <TableRows>
    <TableRow>
      <Height>.25in</Height>
      <TableCells>
        <TableCell>
          <ReportItems>
            <Textbox Name="TXTCOMPANY">
              <Value>=Fields!Company.Value</Value>
            </Textbox>
          </ReportItems>
        </TableCell>
        <TableCell>
          <ReportItems>
            <Textbox Name="TXTADDRESS">
              <Value>=Fields!Address.Value</Value>
            </Textbox>
          </ReportItems>
        </TableCell>
      </TableCells>
    </TableRow>
  </TableRows>
</Details>
```

Because we've set the `XmlTextWriter` object's `Formatting` property to `Indented`, the document is properly formatted. When done with the RDL file, we force the `writer` object to write any data it contains to the output stream and close the `strm` object to unlock the file we're writing to:

```
writer.Flush()
strm.Close()
```

It takes quite a bit of code, but the process is quite straightforward. If you have good reason to generate reports from within your code, you can write a code generator to simplify the process of the custom report creation.

Publishing the custom report is also trivial and the process was discussed earlier in this chapter. Run the ReportDefinition project and experiment with various report types. Use the RDL file generated by the wizard to find out how the various elements of a report are implemented and then insert the appropriate statements in your VB code.

Embedding Images

The images displayed on the report are usually stored as embedded resources in the RDL file. These are the images you use to decorate the appearance of the report, not the images that are displayed along with the report data (such as the product images). The Report Server supports the common image formats (BMP, JPEG, JPE [truncated JPE], GIF, and PNG formats). To embed an image in the report, you can drag an `Image` report item from the Toolbox and drop it onto a report. The designer will display the Image Wizard (shown in Figure 13.4), where you can select one or more images to include in your report.

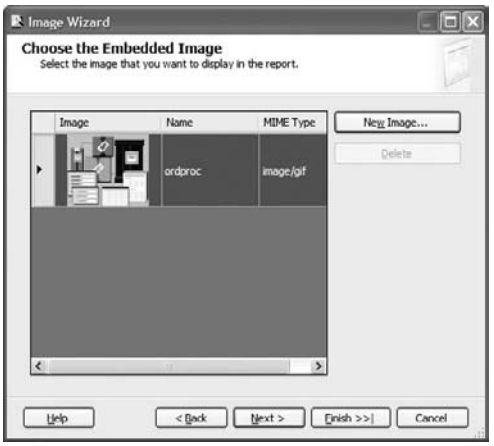


FIGURE 13.4
Embedding an image resource to the report

Embedded images are encoded in Multipurpose Internet Mail Extensions (MIME) format; they're inserted into the report as long strings. The rendering engine reads the image definition and renders it on the report, always in the appropriate report item. You can also add an image to the project, in which case the image file is copied as a resource file to a Report Server during deployment. If you add a new image as an embedded resource to a project and then examine the RDL file, you'll see that the designer has inserted the following statement in the file:

```
<EmbeddedImages>
      <EmbeddedImage Name="imgCustomers">
        <ImageData>
          cywNpBt3K1gaqW$\ldots$90AVAfvgTB/3nJE4IAAA7
        </ImageData>
        <MIMEType>image/gif</MIMEType>
      </EmbeddedImage>
    </EmbeddedImages>
```

The `EmbeddedImages` element appears in the `Report` section of the file. The `Name` attribute of the `EmbeddedImage` tag allows you to refer to the image by its name later in the document. For the time being, we've only embedded an image to the file; we haven't used it yet. In the `ImageData` element, you will see a long string of characters (I've used an ellipsis to indicate that there is much more data than shown here). This string is the representation of the image data. In effect, they're binary data stored in Base64 format.

One way to get the Base64 description of an image's data is to insert the image into a report using the visual tools and then copy the `EmbeddedImages` element into your RDL file. You can also create the same string programmatically. To do so, you must read the image file's contents into an array of bytes and convert them to a Base64 string with the `Convert.ToBase64String` method. Here's the code segment that stores the Base64 representation of a GIF file to a string variable:

```
Dim bytes() As Byte
bytes = File.ReadAllBytes( _
            "c:\Windows\System32\oobe\Images\Hand1.gif")
Dim strImage As String = Convert.ToBase64String(bytes)
```

(I've used an image that should exist on the default installation of a Windows system; feel free to replace it with a more suitable one).

Now that you have a string with the image's data, you can embed it in the RDL file by adding an `EmbeddedImages` element to the document. This element should appear right after the `DataSets` element and it's implemented with the following statements:

```
writer.WriteStartElement("EmbeddedImages")
writer.WriteStartElement("EmbeddedImage")
writer.WriteAttributeString("Name", Nothing, "imgCustomers")
writer.WriteElementString("ImageData", strImage)
writer.WriteElementString("MIMEType", "image/gif")
writer.WriteEndElement() ' EmbeddedImage
writer.WriteEndElement() ' EmbeddedImages
```

To use the embedded image, you must insert an <`Image`> tag in the `ReportItems` element of the document. The <`Image`> tag must have a `Name` attribute and several nested tags that determine the size and scaling of the image. To use an image in the report, you must insert an `Image` element in the appropriate section of the report (the `Body`, `Header`, or `Footer` element). The following statements insert the embedded image at the top of every page in the report, and they should appear in the `ReportItems` element of the report's `Body` element:

```
writer.WriteStartElement("Image")
writer.WriteAttributeString("Name", Nothing, "image1")
writer.WriteElementString("Sizing", "AutoSize")
writer.WriteElementString("Left", "0.19048cm")
writer.WriteElementString("Top", "0.31746cm")
writer.WriteElementString("ZIndex", "2")
writer.WriteElementString("Width", "2.5cm")
writer.WriteElementString("Source", "Embedded")
writer.WriteStartElement("Style")
writer.WriteEndElement()     ' Style
writer.WriteElementString("Height", "2.5in")
writer.WriteElementString("Value", "imgCustomers")
writer.WriteEndElement()     ' Image
```

The `Image` element is a report item, just like the `Textbox` item, and it displays an image. No particular style has been applied to the `Image` report item, but you must specify the item's location

and size carefully so that it won't interfere with the report's data. The rendering engine will scale the image to fit in the specified area. Yet you shouldn't embed images just because they will be scaled automatically on the `Image` report item.

Exploring the RDL Elements

In the last section of this chapter you'll find a quick overview of the elements that make up the Report Definition Language and their structure. The top element in the hierarchy is `Report`, which is the container for all other elements. There are a number of elements that are common to all reports, such as the elements that define the report's data sources and data sets, the report's parameters, and more. After that you will find the details of the `Table` element, which is used to produce tabular reports. There are similar elements for the `Matrix` element, which weren't included in this list. You can find the complete RDL specifications in a document published by Microsoft, at this URL: `http://www.microsoft.com/sql/technologies/reporting/rdlspec.mspx`.

The *Report* Element

The `Report` element contains information about the report, which is the top-level element of the report definition file. The elements you can nest under the `Report` element are the following:

`Author` The name of the report's author

`AutoRefresh` The rate at which the report page is automatically refreshed, in seconds

`DataSources` The data sources from which data sets will be populated

`DataSets` The data to be displayed by the report

`Body` The report's body

`ReportParameters` The report's parameters

`Custom` Custom information to be passed to the report engine

`PageHeader` and `PageFooter` The header/footer to be inserted at the top of each page of the report

`PageHeight` and `PageWidth` The page's dimensions expressed in inches

`LeftMargin`, `RightMargin`, `TopMargin`, `BottomMargin` The page's margins

`EmbeddedImages` The images that are embedded into the report definition file

`DataTransform` The path to an XSLT file with the transformation to be applied to the report's data (a string like `"/xsl/trData.xsl"`)

The *EmbeddedImage* Element

The `EmbeddedImage` element defines an image embedded within the RDL file and may contain the following elements:

`Name` The image's name in the report's context

`MIMEType` The image's `MIMEType`, which is one of the following strings (`"image/bmp"`, `"image/jpeg"`, `"image/gif"`, `"image/png"`, `"image/xpng"`)

`ImageData` The image's data encoded in Base-64 format

The *ReportParameters* Element

The `ReportParameters` element contains a list of parameters for the report and may contain one or more `ReportParameter` elements.

The *ReportParameter* Element

This element is the definition of a report parameter and may contain the following elements:

`Name` The parameter's name.

`DataType` The parameter's data type (`Boolean`, `DateTime`, `Integer`, `Float`, `String`).

`Nullable` If true, the parameter can have a `Null` value.

`DefaultValue` The parameter's default value.

`AllowBlank` If true, the value for this parameter can be an empty string.

`Prompt` The prompt to be used for the parameter value. If omitted, the user won't be prompted for this parameter.

`Hidden` If true, the parameter is not displayed to the user.

`ValidValues` The valid values for the parameter.

`MultiValue` If true, the parameter's value comes from a set of predefined values.

The *ParameterValue* Element

This element is a parameter value and may contain the following elements:

`Value` One of the possible parameter values.

`Label` The values that will be displayed to the user. If omitted, `Value` is used as the label.

If the parameter is a multivalued one, set the `MultiValue` element to true and insert a `ValidValues` element that contains a reference to the `DataSet` with the parameter's possible values.

The *DataSet* Element

The `DataSet` element contains information about the report's `DataSet` and may contain one of the following elements:

`Name` The `DataSet`'s name.

`Fields` The `DataSet`'s fields. This element is made up of `Field` elements.

`Query` Identifies the query that populates the `DataSet`.

`CaseSensitivity` If true, the data is case sensitive (used for searching and sorting).

`Collation` The locale to be uses for the collation sequence (used for sorting). Use the standard Microsoft SQL Server collation codes (`en-us` for American English).

`AccentSensitivity` If true, the data is accent sensitive (this setting applies to languages with accent marks, like French or Greek).

The *Fields* Element

The `Fields` element specifies the fields that will be included in the report and is made up of `Field` elements.

The *Field* Element

The `Field` element contains information about the fields of the `DataSet` and may contain the following elements:

`Name` The field's name within the report (need not be the same as the database field name)

`DataField` The name of the field in the query

`Value` The value of a calculated field, such as `= Fields!Total.Value + Fields!Freight.Value`

The *Query* Element

The `Query` element contains the description of the query that will be executed to retrieve the data for the report and may contain the following elements:

`DataSourceName` The name of the data source against which the query will be executed.

`CommandType` The type of the query (set it to `Text` for SQL statements, `StoredProcedure` for stored procedures, or `TableDirect` to select all rows of a table).

`CommandText` The `SELECT` statement that will be executed to retrieve the data. If the `CommandType` element is `TableDirect`, the `CommandText` element should be set to the name of the table.

`QueryParameters` The query's parameters. This element is made up of `QueryParameter` elements.

`Timeout` The timeout (in seconds) for the query.

The *QueryParameters* Element

The `QueryParameters` element contains the query's parameters and is made up of `QueryParameter` elements.

The *QueryParameter* Element

The `QueryParameter` element contains information about a query parameter, and the following elements may appear within the `QueryParameters` element:

`Name` The parameter name

`Value` The value of the query parameter

The *DataSources* Element

The `DataSources` element contains information about the connection to the data source and is made up of `DataSource` elements.

The *DataSource* Element

The `DataSource` element contains information about a data source and may contain the following elements:

`Name` The data source's name.

Transaction If true, the DataSets that are based on this data source should be executed in the context of a single transaction.

ConnectionProperties Information about the connection string to be used to connect to the database. The ConnectionProperties element may contain the following elements:

DataProvider The type of the data source (SQL, OLEDB-MD, OLEDB, ODBC, Oracle, DB2, XML).

ConnectString The connection string for the data source.

IntegratedSecurity If true, the Report Server will connect to the data source using integrated security.

Prompt The prompt to be displayed to the user when prompting for database credentials.

The *Body* Element

The Body element contains report items and the visual elements of the reports. This element has one or more ReportItems and may contain any of the following elements:

ReportItems This element defines the area that contains the various elements of the report (text boxes, images, regions, and so on).

Height The height of the report's body.

Columns The number of columns for the report.

ColumnSpacing The spacing between columns in the report. The default value is 0.5 in.

Style Style information for the report's body.

The *PageHeader* Element

This element defines the items that will appear at the top of every page of the report and may contain the following elements:

Height The height of the header.

PrintOnFirstPage Set it to True to display the header on the first page of the report.

PrintOnLastPage Set it to True to display the header on the last page of the report.

ReportItems This element defines the items of the header and is made up of text boxes, images, lines, and rectangles.

Style Style information for the header.

The *PageFooter* Element

This element defines the items that will appear at the bottom of every page of the report and may contain the following elements:

Height The height of the footer.

PrintOnFirstPage Set it to True to display the footer on the first page of the report.

PrintOnLastPage Set it to True to display the footer on the last page of the report.

ReportItems This element defines the items of the footer and is made up of text boxes, images, lines, and rectangles.

Style Style information for the footer.

The *Table* Element

The Table element defines a tabular report and may contain the following elements:

TableColumns The table's columns.

Header The table's header row(s).

TableGroups The table's groups.

Details The table's detail row(s).

Footer The table's footer row(s).

FillPage If true, the table should expand to fill the page.

The *TableColumns* Element

The TableColumns element defines the columns in the table and must contain at least one TableColumn element.

The *TableColumn* Element

The TableColumn element defines a column in the table and may contain the following elements:

Width The column's width.

Visibility Set it to False to hide the column.

FixedHeader If true, the current column will be displayed on the page even when the user scrolls the page horizontally. The fixed header columns include either the first or the last column and must be contiguous.

The *Header* Element

The Header element defines the header of a table or group within the table. It contains a TableRows element and may contain the RepeatOnNewPage or FixedHeader element:

TableRows The rows that make up the header.

RepeatOnNewPage Indicates whether the footer should be repeated on each page of the same table or group. If false, the footer will appear on the first page of the group.

FixedHeader If true, the header will remain at the top of the page, even when the user scrolls down the page. This element is valid for table headers only.

The *Footer* Element

The Footer element defines the footer of a table or a group within the table. It contains a TableRows element and may contain a RepeatOnNewPage element:

TableRows The rows that make up the footer.

RepeatOnNewPage Indicates whether the footer should be repeated on each page of the same table or group. If false, the footer will appear on the first page of the group.

The *TableGroups* Element

The `TableGroups` element defines the groups in the table and contains one or more `TableGroup` elements.

The *TableGroup* Element

The `TableGroup` element defines a single group in the report's `Details` section. It contains one `Grouping` element and may contain one or more of the following elements:

Grouping An expression (field name) by which the data will be grouped.

Sorting An expression (field name) by which the data will be sorted.

Header A group's header.

Footer A group's footer.

Visibility Set it to False to hide the group.

The *Details* Element

The `Details` element defines the data that will be displayed on the report's grid area and may contain the following elements:

TableRows The details rows for the table. Each `Details` element must have exactly one `TableRows` element.

Grouping One or more expressions (field names) accoording to which the detail data will be grouped.

Sorting The expression according to which the detail data will be sorted.

Visibility Set it to False to hide the detail section.

The *TableRows* Element

The `TableRows` element defines a list of table rows and contains one or more `TableRow` elements.

The *TableRow* Element

The `TableRow` element defines a row of cells in the report's details section and may contain the following elements:

TableCells Each row of cells is represented by a `TableCells` element, which in turn contains a `TableCell` element per column.

Height The row's height in inches.

Visibility Set it to False to hide the row.

The *TableCells* Element

The `TableCells` element defines the contents of an entire row in the `Details` section and consists of `TableCell` elements, one per field.

The *TableCell* Element

The `TableCell` element defines the contents of a single cell. It contains a single `ReportItems` element and may contain a `ColSpan` element:

`ReportItems` Each `TableCell` element contains an item such as a `Textbox`, `Rectangle`, and so on. The `ReportItems` collection of the `TableCell` object must contain exactly one `ReportItem`.

`ColSpan` If you want a specific cell to span across mutliple columns, set this element to the appropriate value. There's no need to specify a `ColSpan` setting of 1.

The *Textbox* Element

The `Textbox` element defines a rectangular region for displaying text and may contain the following elements:

`Value` The text to be displayed; it's usually a field's name.

`CanGrow` If true, the `Textbox` element grows to accommodate the size of the text.

`CanShrink` If true, the `TextBox` element shrinks to match the size of the text.

`HideDuplicates` Indicates that the value should not be displayed when it's the same as the value of the preceding instance of the same element (the same cell on the previous row).

`UserSort` Indicates that an end-user sort control should be displayed as a part of this text box.

`DataElementStyle` Indicates whether text box value should render; it can have one of the following values: `Auto`, `AttributeNormal`, or `ElementNormal`.

The *Image* Element

The `Image` element is a report item for displaying images and may contain the following elements:

`Source` Identifies the source of the image and can have one of these values: `External` (the path or URL of the image), `Embedded` (the name of the `EmbeddedImage` element in the RDL file), or `Database` (the value is a database field).

`MIMEType` Identifies the image's MIME type and can have one of the following values: `image/bmp`, `image/jpeg`, `image/gif`, `image/png`, `image/x-png`.

`Sizing` Determines how the image will be sized on the `Image` element. It can have one of the following values: `AutoSize` (the item is resized to accommodate the image), `Fit` (the image is resized to match the size of the image element), `FitProportional` (the image is resized preserving its aspect ratio), or `Clip` (the image is clipped).

The *Style* Element

The `Style` element contains information about the style of a report item. It may contain a number of subelements, whose names match the standard Cascading Style Sheet properties. You can easily understand how to use these properties by their names. Here are the elements you can nest within a `Style` element, and they apply to all report items that recognize the `Style` element:

`BorderColor`, `BorderWidth` The border's color and width.

BorderStyle The border's style (`None`, `Dotted`, `Dashed`, `Solid`, `Double`, `Groove`, `Ridge`, `Inset`, `WindowInset`, `Outset`).

BackgroundColor, Color The item's background and foreground colors.

FontStyle The style of the item's font (`Normal`, `Italic`).

FontFamily The family of the item's font.

FontSize The size of the item's font.

FontWeight The weight of the item's font (`Lighter`, `Normal`, `Bold`, `Bolder`).

TextDecoration The special style of the item's font (`Underline`, `Overline`, `LineThrough`, `None`).

TextAlign The horizontal alignment of the text in the item (`Left`, `Center`, `Right`, `General`).

VerticalAlign The vertical alignment of the text in the item (`Top`, `Middle`, `Bottom`).

PaddingLeft, PaddingTop, PaddingBottom, PaddingRight The padding on the four edges of the item. These properties are expressed as number of points.

WritingMode Indicates whether the text in the item is written horizontally or vertically.

Language The language for the report's text.

Calendar The calendar for formatting the report's dates (`Gregorian`, `English`, `Japanese`, and so on).

Global Variables

While designing a report, either with the visual tools or with VB (or any `.NET` language) code, you have at your disposal a few useful global variables, which you can use in any section of the report:

PageNumber The current page number (this variable is available only in the page header/footer of the report, not in the report's body).

TotalPages Total number of pages in the report (this variable is available only in the page header/footer of the report).

ExecutionTime The date and time the report began executing.

ReportFolder The path of the report on the server (something like `Customers`).

ReportName The name of the report.

The Bottom Line

Understand the report definition schema.

Master It To create reports from within your custom application's code, you must understand the structure of a report.

Define a report's data source and data sets.

Master It Most reports use a single data source and one or more `DataSets`. Data sources and `DataSets` can be specified in RDL and embedded in an RDL file, which you can then publish in a Report Server folder. How do we specify the report's data in RDL?

Create report and query parameters.

Master It Most reports use parameters, and most of the report parameters are passed as arguments to the stored procedures or SQL statements that populate the `DataSet` with the report's data. You should be able to create report and query parameters from within your code.

Use the `ReportItems` element.

Master It What's a `ReportItem` and where does it appear in an RDL file?

Use the `XmlTextWriter` class.

Master It The `XmlTextWriter` class can automate the generation of XML files. Which are the basic methods of the `XmlTextWriter` class for inserting tags and attributes to an XML file?

Chapter 14

Report Server Security

In Chapter 3, we compared protecting information to protecting a building that you own. Only your employees should have access to enter your building and use your resources. Once inside, employees should not have access to everything. Only accountants should have access to the accounting department, and almost no one should have access to your office, so you put various security systems in place like alarm systems and locks on the doors.

We also explored how to lock down resources stored on your Reporting Services server using role-based security. In this chapter, we will discuss how to give users access to Reporting Services, or give them a ''key'' to get in the ''building.''

Of course, it is not as simple as just giving your users a login and letting them run free. You need to understand how security works in Reporting Services so that you can choose the proper security methods to restrict access to certain parts of the system. For example, you may want to limit the ability to view or modify a particular report to certain people. You may want to secure either an individual report or an entire folder of reports, snapshots, and models. You will also find some techniques to restrict data within a report based on the user currently accessing the report.

NOTE

In this chapter I assume that you are a local system administrator on the Report Server.

In this chapter you will:

- Explore the various types of authentication
- Learn how to set up SSL
- Understand how to secure folders, models, and My Reports
- Learn how to mitigate injection attacks

Authentication

Reporting Services security is provided by a combination of `ASP.NET`, Internet Information Services (IIS), and Windows security. Combined, these components ensure that only authorized users gain access to the system.

`ASP.NET` security provides application-level security for both the Report Server and Report Manager. IIS security controls access to Report Manager and the Report server virtual directory (the SOAP endpoints of the Report Server). IIS also authenticates user connections to a Report Server instance.

These components perform two key functions in the security process: authentication and authorization. In this section, you will learn how to configure IIS to perform these functions.

Types of Authentication

Some people try to hack their way into secured systems, like Reporting Services, by trying to log in using another person's identity. So it is necessary for the system to make certain that all users are who they claim to be. *Authentication* is the process that proves that a user is who they say they are. This process typically involves asking a user for a username and password by presenting them with a dialog box. There are three authentication options that you can use with Reporting Services:

- Integrated Windows authentication is the default authentication method for the Report Server and Report Manager virtual directories. You'll learn more about this a little later in this chapter.
- You can also use Anonymous access, but only if you are using a custom security extension. This is because anonymous access limits your ability to vary role assignments because all users will access the Report Server under the Anonymous user account. Anonymous access also prevents users from administering the Report Server through Report Manager.
- You should only use Basic authentication when you are using Secure Sockets Layer (SSL) because Basic authentication sends usernames and passwords to the server in clear text. Anyone with a network sniffer could read the usernames and passwords right off the network if they are not secured using SSL.

NOTE

Digest and Passport authentication methods are also available, but they are not supported by Reporting Services.

Integrated Windows Authentication

The default security setting for Reporting Services is Integrated Windows authentication with no other options checked in IIS. This method works well if all of the users have a Windows account in a single domain or a trusted domain. Those accounts must have access to the web server that hosts Reporting Services, and they must be assigned to a role that gives them access to Reporting Services.

Integrated Windows authentication does not work well if you need to grant access to Reporting Services over the Internet or an extranet. In such cases, you will have to develop a custom authentication scheme by creating a custom security extension.

Regardless of whether authentication is by Windows or a custom method, all users log on to the server using the `LogonUser` method, a part of the Reporting Services web service that is used to pass user credentials to a report server for validation. The web service uses HTTP headers to pass a cookie that contains an authentication ticket from the server to the client for validated logon requests. Here is how it works:

1. A client application calls the web service method `LogonUser` to authenticate a user.
2. The web service makes a call to the `LogonUser` method of your security extension, specifically the class that implements `IAuthenticationExtension`.

3. Your implementation of `LogonUser` validates the username and password in the user store or security authority.
4. Upon successful authentication, the web service creates a cookie and manages it for the session.
5. The web service returns the authentication ticket to the calling application on the HTTP header.

When the web service successfully authenticates a user, it generates a cookie that is used for subsequent requests. The cookie is returned from the `LogonUser` web service method and is used in subsequent web service method calls and in URL access initiated by the same session.

Forms Authentication

Integrated Windows authentication works great for users who have Windows accounts, but Unix, Mac, Novell, Linux, and other users may not have Windows accounts. You can use Forms Authentication to support these users.

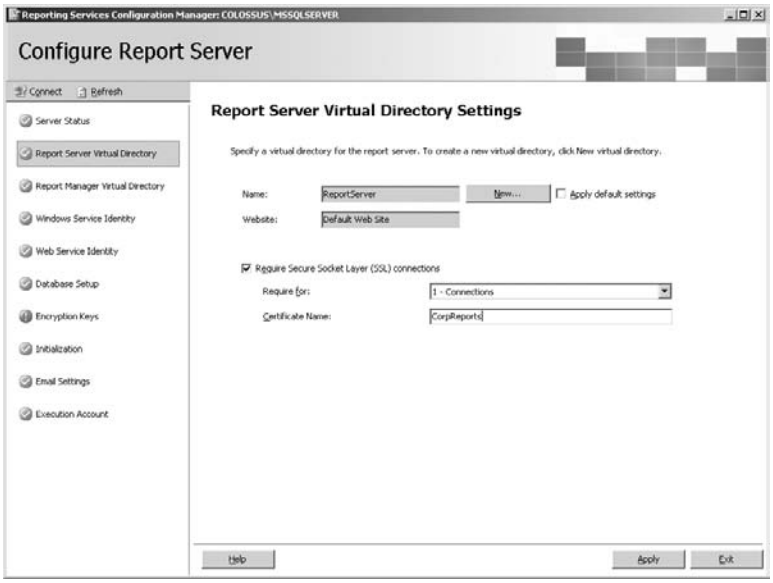
The Forms Authentication method directs a request from an unauthenticated user to an HTML form, where they are prompted to enter a username and password. Once the user provides credentials, the system issues a cookie containing an authentication ticket. For subsequent requests, the system will check the cookie first to see if the user was already authenticated by the Report Server.

If you extend Reporting Services to use Forms Authentication, always use SSL for all communications with the Report Server. This will prevent a malicious user from gaining access to another user's cookie, because all data sent from a client through an SSL connection is encrypted so that malicious users cannot intercept passwords or data sent to a Report Server.

Configuring Report Server for SSL

SSL provides an extra layer of security by encrypting all the data that is transferred between the client and the server. The data is encrypted using a certificate that is installed on the website in IIS. You can create a certificate using Certificate Services in Windows, but it is better to get a certificate from a third party, such as VeriSign, because third parties are more widely recognized. You can specify SSL connection levels for Report Server virtual directories through the Reporting Services Configuration tool or by modifying the configuration settings. Once you have the certificate, follow these steps to configure Reporting Services to use SSL:

1. Assign a certificate to your Default Web Site in IIS Manager. Open IIS Manager and expand your server; then expand Web Sites, right-click Default Web Site and select Properties, and on the Directory Security tab click the Server Certificate button and follow the instructions.
2. Click Start ➢ Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ Reporting Services Configuration.
3. Specify the machine name and instance name you want to connect to and click Connect.
4. Click the Report Server Virtual Directory link on the left side of the page.
5. Check the box next to Require Secure Socket Layer Connections.
6. In the Require For drop-down box, select 1-Connections.
7. Type the fully qualified name of the server in the Certificate Name box.



8. Click Apply. A task status window appears with a report updating the system.
9. Click Exit.

In the Require For box in step 6 you may have noticed that there are three options to choose from. They are described in Table 14.1.

TABLE 14.1: SecureConnection Levels

LEVEL	DESCRIPTION	WEB METHODS REQUIRING SSL
1	Minimally secure. This rejects all web service calls that are made over an insecure connection and that might pass sensitive data such as user credentials. However, it is considered insecure since sensitive data can still be sent by the client to the Report Server and exposed before the Report Server handles the request and rejects it.	`Render` (when the credential setting for the report that is being rendered is set to prompt), `CreateDataSource`, `GetDataSourceContents`, `SetDataSourceContents`, `GetReportDataSources`, `SetReportDataSources`, `CreateReport`, `GetReportDefintion`, `SetReportDefinition`, `CreateDataDrivenSubscription`, `SetDataDrivenSubscriptionProperties`, `GetDataDrivenSubscriptionProperties`, and `PrepareQuery`
2	Secure. All rendered reports and all web service calls require a secure connection.	All Level 1 methods, `Render` (all calls), and `RenderStream`
3	Most secure. All calls made to the Reporting Services SOAP API require a secure connection.	All

If you decide to use SSL and set the security connection level to 3-All Soap APIs, you need to modify the <UrlRoot> configuration setting in the RSReportServer.config file to compensate.

Reporting Services uses the <UrlRoot> configuration setting to resolve links in e-mail messages to reports on a Report Server. When you use the Reporting Services Configuration tool to deploy a Report server on a computer that has an SSL certificate installed, the value for <UrlRoot> is set to HTTP, but it needs to be HTTPS. As a result, users who click report links in e-mail messages will get a broken link message rather than the report they expect.

To correct this issue, you must manually edit the RSReportServer.config file to correct the <UrlRoot> value. The format of the URL is as follows:

```
<UrlRoot>https://server/virtualdirectory</UrlRoot>
```

When editing RSReportServer.config, be sure to specify the same values that you typed in the Report Server Virtual Directory page in the Reporting Services Configuration tool.

Changing Service Accounts and Passwords

Reporting Services uses a Windows service and web service to support server operations, both of which use an account to access resources that they need to perform their jobs. Over time, you might need to update passwords, accounts, or both.

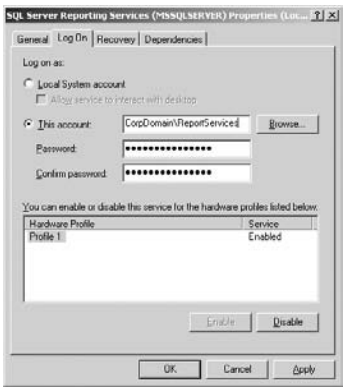
The account for the Report Server web service is established when you configure the Report Server virtual directories and is based on the version of IIS you are using. If you are running IIS 5, the Report Server web service account runs as the ASP.NET process identity. You cannot configure the Report Server web service to run under a different account.

If you are using IIS 6.0 and you installed Reporting Services in the default configuration, the web service is controlled by the application pool, which has the NetworkService identity assigned by default on Microsoft Windows Server 2003. On Windows 2000 Server and Windows XP, ASP.NET runs under its own account as *computername*\ASPNET.

The Report Server Windows service is configured during Setup. You can use the Reporting Services Configuration tool to modify the service account properties after Setup is finished.

If the Report Server Windows service runs under a domain account and the password expires, the service will be unavailable until you specify a new password. Follow these steps to reset the password:

1. Click the Start menu, point to Control Panel, point to Administrative Tools, and click Services.
2. Right-click SQL Server Reporting Services and select Properties.
3. On the Log On tab, type the new password in the Password and Confirm Password text boxes.

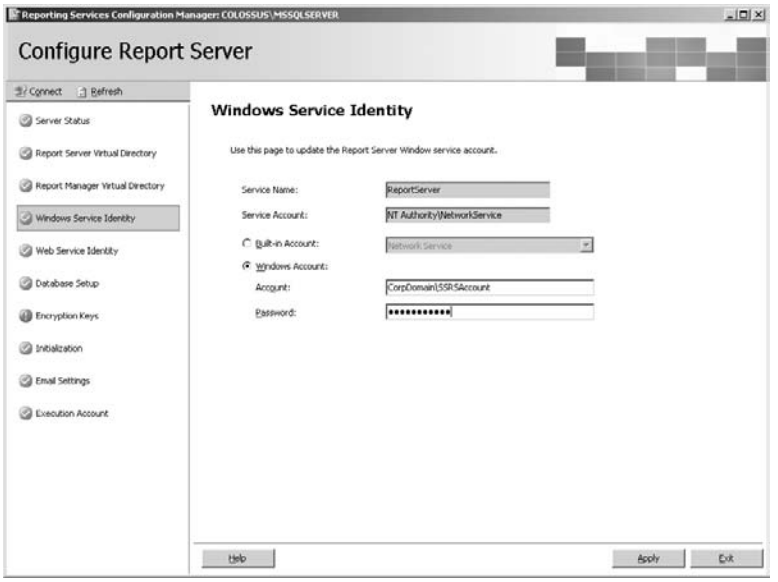


Changing the password is fairly straightforward. Changing the account can affect the Report Server operations, and even crash the server. Be sure to use the Reporting Services Configuration tool to avoid that outcome. This tool performs the following additional steps to ensure the Report Server remains available:

- Automatically updates the encryption key to include the profile information of the new account.
- Automatically updates the login permissions on the SQL Server Database Engine used to host the Report Server database.
- Automatically adds the new accounts to the Report Server group created on the local computer. This group is specified in the Access Control Lists (ACLs) that secure Reporting Services files.

Use the following steps to reconfigure the Report Server Windows service account:

1. Start the Reporting Services Configuration tool and connect to the Report Server.
2. On the Windows Service Identity page, specify the new account and password.



3. Click Apply.
4. When prompted to back up the symmetric key, type a password and file name for the symmetric key backup, and click OK.

Each step in the service account update is recorded in the tasks panel at the bottom of the page. The Report Server Windows service will be stopped and restated during the account update.

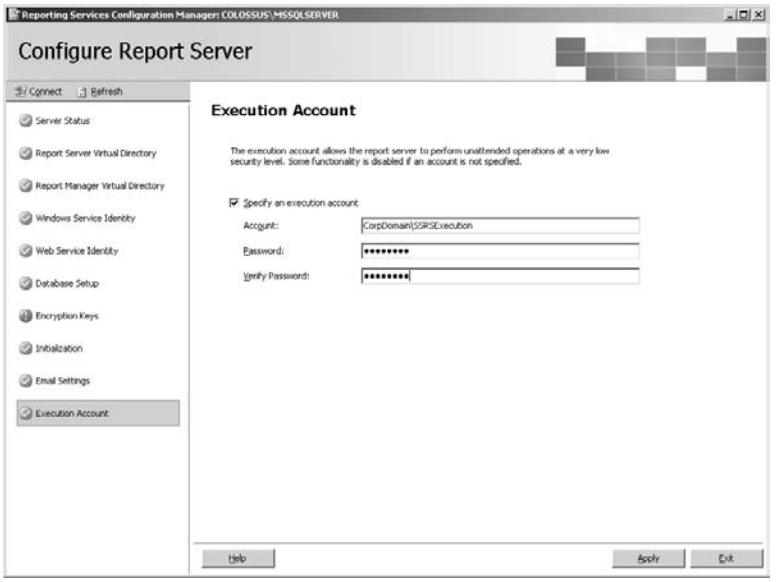
Once done, make sure that you verify that the account has been reset properly by opening a report using stored credentials.

Configuring an Account for Unattended Report Processing

Unattended report processing refers to any report execution process that is triggered by an event (either a schedule-driven event or data refresh event) rather than a user request. The credentials of the Report Server Microsoft Windows service account or web service account are never used to connect to other computers, so the Report Server requires an unattended report processing account to log on to the computer that hosts the external data source.

Follow these steps to create the account using the Reporting Services Configuration tool:

1. Start the Reporting Services Configuration tool and connect to the Report Server instance you want to configure.
2. On the Execution Account page, select Specify An Execution Account.
3. Type the account and password, retype the password, and then click Apply.



You can also use the `rsconfig.exe` utility with the `-e` argument, which directs the utility to write the account information to the configuration file. Follow these steps to configure the account using `rsconfig.exe`:

1. Create or select a domain account that has access to computers and servers that provide data or services to a Report Server.
2. Open a command prompt and execute the following command:
   `rsconfig -e -u` *`domain\username`* **`-p`** *`password`*`.`

Authorization

After an account is authenticated, it must be authorized each time it tries to access a resource. Authorization is the process of determining whether an identity should be granted the requested type of access to a given resource in the Report Server database. Reporting Services authorization

is role based and is controlled by the Report Server's security extension. The process looks like this:

1. After a user is authenticated, the client application makes a request to the Report Server via Reporting Services web service methods. An authentication ticket is then sent to the Report Server as a cookie in the HTTP header of each web request.
2. The cookie is validated prior to any access check.
3. Once the cookie is validated, the report server calls `GetUserInfo` and the user is given an identity.
4. The user attempts an operation and the Report Server calls the `CheckAccess` method.
5. The security descriptor is retrieved and passed to the extension implementation of `CheckAccess`. The user, group, or computer is compared to the security descriptor of the item being accessed and is authorized or not authorized to perform the requested operation. If authorized, the web service performs the operation and returns a response to the calling application.

We discussed role-based security in detail in Chapter 3, but a brief refresher is in order here to help you better understand authorization.

A role definition is a grouping of permissions that are often required together for a specific user task (such as publishing a report). A role assignment maps a specific role to a user or group and a securable object, such as a folder or report. The following objects are securable in Report Server:

- Folders
- Reports
- Report models
- Resources
- Shared data sources

Report Server provides both default roles and default role assignments, which together create a basic security model that meets most needs. The default role assignments protect the System object and the Home folder of the folder hierarchy. As you create new folders within the Home folder, they inherit the default settings. You can expand this basic security model by adding specific users to specific roles, and by using additional roles. Report Server has the following predefined roles:

Browser Role Run reports and navigate through the folder structure

Content Manager Role Define a folder structure for storing reports and other items, set security at the item level, and view and manage the items stored by the server

Publisher Role Publish content to a Report Server

My Reports Role Build reports for personal use or store reports in a user-owned folder

System Administrator Role Enable features and set defaults, set sitewide security, define create role definitions, and manage jobs

System User Role View the schedule information in a shared schedule, or view other basic information about the Report server

Report Builder Build and edit reports in Report Builder

The seven default roles provided with Reporting Services will meet most of your security requirements. In circumstances where they do not, Reporting Services roles can be modified, including the predefined roles.

The recommended method to manage security is to supplement the default security setting with a minimum set of role assignments that provide users access to reports. In this section, you'll see how to add role assignments and use them to secure different parts of the system.

Securing Folders

The most effective method for authorizing users to access your Report Server content is to place the content in folders and secure the folders. This method gives you a great deal of flexibility in your design.

For example, you could create a staging area for reports that only developers can access. You can do this by creating a folder that developers use to test reports before moving them to a final location. Then, to control access to this area you can define one role assignment that only allows report authors to add and delete items, and a second role assignment that allows testers to run reports but not to add or remove items. Because the role assignments are defined explicitly for testers and report authors, no other users (except for local system administrators) can access the folder.

Security is inherited through the folder hierarchy, starting with the root node of the folder hierarchy, the Home folder. Because security is inherited, it is advisable to set a fairly restrictive security policy on the Home folder. The Browser role is ideally suited for this purpose because it provides view-only access.

The following tasks should be used to assign customized access for modified or new role assignments.

View Folders View the folder hierarchy and read-only properties that indicate when the folder was created and modified. Users cannot view items in the folder unless role assignments also include View Reports, View Models, View Resources, and View Data Sources.

Manage Folders View folder properties, change the name or description, or move the folder to another location and create folders.

Manage Reports Add reports from the filesystem to a folder and publish reports from Report Designer to the Report Server.

Manage Data Sources Add new shared data source items to a folder and change existing shared data sources.

Set Security on Items Create and modify role assignments that control access to the folder. This task must be used with either View Folders or Manage Folders. Otherwise, it has no effect since the user will not be able to select the item.

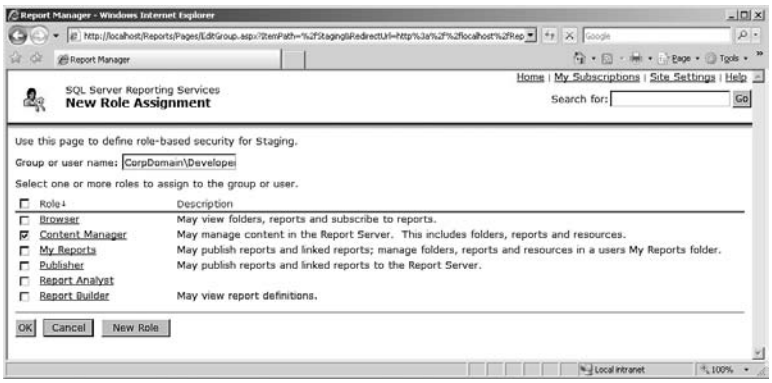
Applying Item Security

When you apply security to a folder, the items within that folder and its nested folders inherit those security settings. For example, if you take away the ability to view a report or other item, the user will not even see that item on the Contents page. Whenever possible, organize content by folders so that you can take advantage of this functionality and minimize the administrative overhead of managing security for individual items.

If necessary, you can override security on individual items by adding role assignments to reports, resources, or data sources. After choosing to override security on an item, you'll have the option to restore the parent folder's security settings when needed.

In this procedure, you'll create a folder named `Staging` and restrict access to the folder by adding a role assignment for the Users group to this folder.

1. Add a Windows Group named Developers In Computer Management and add your account as a member.
2. Open Report Manager in Internet Explorer at `http://localhost/Reports`.
3. Click the New Folder button.
4. In the Name text box, enter **Staging**.
5. Click OK.
6. Click the Staging link and switch to the Properties tab.
7. Click on the Security link.
8. Click the Edit Item Security button.
9. Click OK when asked if you want to override security settings.
10. Click New Role Assignment.
11. In the User or group name box, enter ***domain*\Developers**.
12. Check the box next to Content Managers.



13. Click OK.

Securing Models

As discussed in Chapter 8, report models are used as data sources to create and use ad hoc reports. Report models can be secured through the folder hierarchy, item security, and database security.

Securing a Report Model in the Folder Hierarchy

As with all items that are stored on a Report Server, you can define item-level role assignments that determine whether a user can view or manage a report model. The tasks necessary to use a model are listed in Table 14.2.

TABLE 14.2: Model-Related Tasks

TASK	DESCRIPTION
Manage Models	Create, view, and delete models, and modify model properties.
Browse Models	Allows a user to view a portion of a model.
View Models	View models, use models as data sources, and query models.

SECURING ITEMS IN A MODEL

Model item security allows you to control access to specific parts of a model using SQL Server Management Studio. After you enable model item security, you can create role assignments on specific nodes in the model namespace. The parts of a model you can secure in Management Studio are:

- Root node
- Folders
- Entities
- Model roles (which refers to the relationship between entities)
- Fields

When model item security is not enabled, all permissions for viewing the data that the model represents are determined through role assignments on the model and report in the Report Server folder hierarchy. Model item security is off by default.

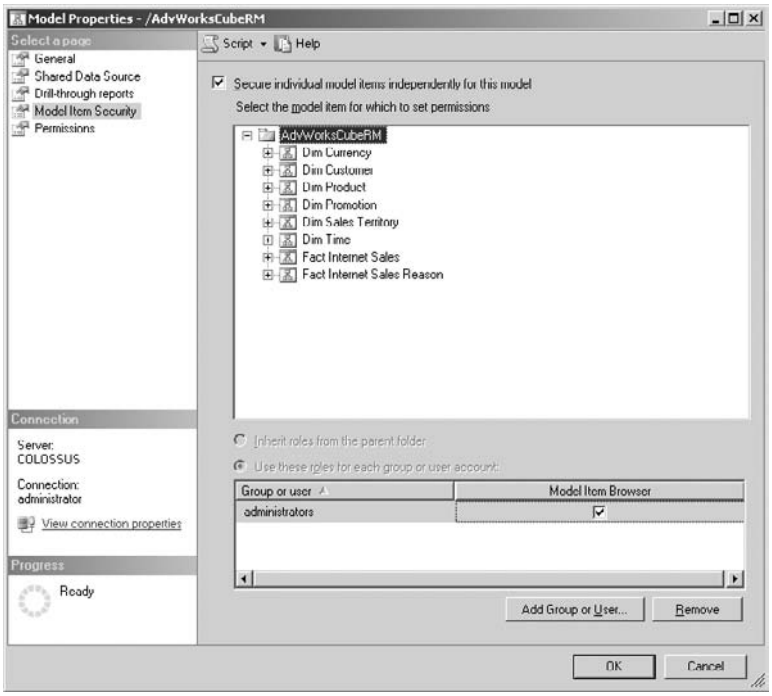
When enabled, model item security is transparent to the user. If a user does not have access to a particular branch of the model hierarchy, they do not see that portion of the model in the report. If you do not want any users to see a model item, just set the Hidden property for the item to true in Model Designer.

Hiding an item does not remove it from calculations or relationships. For example, if you hide a field that is used in an expression it will still be used in the expression while it is hidden. Hiding an item hides it for all users, so if you want to vary visibility and access by user or group, use role assignments instead.

Securing items in a model requires a Report Server connection in Management Studio. To secure model items, the report model must be deployed on the Report Server. Follow these steps to secure items in a model:

1. In Management Studio, connect to the Report Server that stores the model.
2. In Object Explorer, navigate to the model, right-click the model you want to secure, and click Properties.
3. On the Model Properties page, click Model Item Security.
4. Select the Secure Individual Model Items Independently For This Model check box.
5. Select the root node. A role assignment is required on the root node.
6. Click Add Group Or User.

7. Type the name of the group or user for which you are creating the role assignment, and then click OK.



8. Navigate to the next entity, relationship, field, or folder that you want to secure.

9. Repeat steps 6 through 8 for each entity you wish to secure.

Securing Models Through Database Security

Models can also be secured at the database level. You can restrict access to tables and columns in your database. If you include these restricted tables or columns in a model and a user who does not have the proper permissions tries to access them through a model, they will receive an access denied error message.

This is handy as a last line of defense, but make sure you know how this will affect access for all users on your system. It is possible to inadvertently block access to users who may need it, and vice versa.

> **Note**
>
> If you are relying on database security to ensure that only specific users are allowed to read particular tables or column values, be sure that you configure the data source to use Windows Integrated security.

Securing My Reports

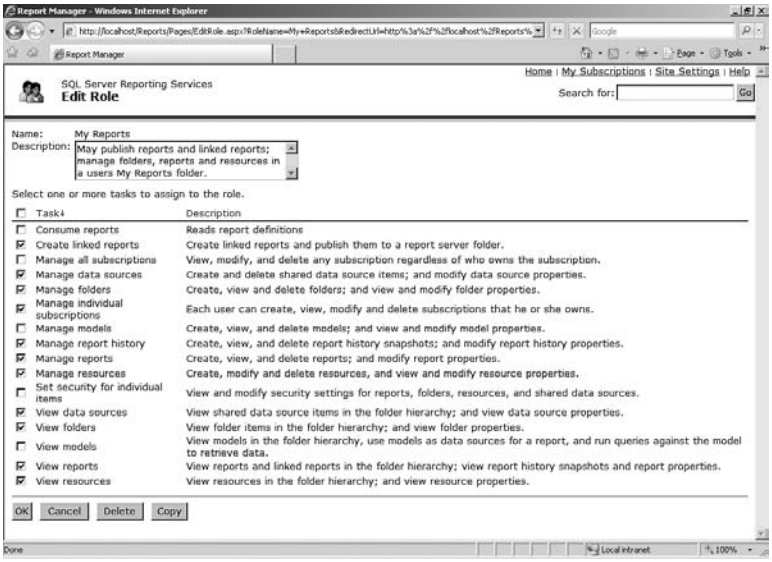
The My Reports feature provides a user-managed workspace for working with reports. The role assignment for My Reports is automatically created for each user who activates a My Reports folder and contains certain preset elements:

- The user's `My Reports` folder, which is located in the `Users Folders\<username>\My Reports` folder.
- The user account, determined when the `My Reports` folder is activated. A folder is activated when a user clicks a `My Reports` folder in Report Manager or when publishing a report to a `My Reports` folder from Report Designer. This folder is also activated when a user requests properties on the My Reports link.
- The predefined role definition for My Reports.

The My Reports role definition is a single-purpose role that contains the tasks required to support content management of a `My Reports` folder.

By default, only Report Server administrators can modify the My Reports role. You can customize the My Reports role by changing the tasks it contains. You can also substitute a different role. Here is how to view the tasks assigned to the My Reports role:

1. Open Report Manager and click the Site Settings link.
2. Click the Configure Item-Level Role Definitions link.
3. Click the My Reports link.



4. Make any changes you may need and click Apply.

Preventing SQL Injection Attacks

If you have a web page that allows users to query a database in some way, like Report Manager, then the web page will have to accept parameters to generate a query. Instead of entering the data to generate a query, a user may be able to enter their own SQL statement and retrieve sensitive data that they were never meant to see. This is called an SQL injection attack, because someone has injected their own SQL query to bypass yours.

For example, suppose you have a website that asks for a name and password. After a user submits the name Joe and the password password1, the following SQL is executed:

```
Select name
from Users
where name = 'Joe'
and pwd = 'password1'
```

This SQL code verifies whether the name and password submitted are correct if a row is returned. The problem occurs if an attacker submits something like the following in place of the password:

```
Mypassword'; select * from users;
```

In this case, when the variables are substituted into the previous SQL statement, this happens:

```
Select name
from Users
where name = 'Tom'
and pwd = 'Mypassword';
select * from users;'
```

The terminator at the end of the password (`;`) does the trick. This tells SQL Server that the original query is done and the next statement is a new query. SQL Server obediently returns the record that your original query asks for, as well as all of the other usernames in the database that the hacker has requested. This could easily be something more destructive, like a `DROP TABLE` or a `DELETE` statement.

This vulnerability is primarily found in applications that have SQL statements hardwired into their code. This is one of the primary reasons that Microsoft recommends you use only functions and stored procedures.

You should develop a policy for all your application development work that seeks to minimize SQL injection vulnerabilities by requiring all code to conform to best practices and using only precompiled modules. If possible, you should specify that dynamic or ad hoc SQL not be allowed.

Reports containing a parameter of type `String` are also vulnerable to SQL injection attacks because when you define a parameter of type `String`, the user is presented with a text box that can take any value, which can be an entry point for malicious code.

To get around this problem you should use an available (or valid) values list so that users cannot enter their own values. Additionally, ensure that any user running the report has only the permissions required to view the data in the report.

Case Study: Talksalot Communications

Talksalot sold phone service for a number of major communications vendors. Each month the vendors would send reports in electronic format detailing the commissions that that were paid and fees that were charged. Everyone in the company was not privy to this information, but some of them wanted to be.

Everyone in the company knew the names of the vendors we worked with, so when I first wrote the report I used a parameter type of string for the vendor name parameter in the report. One day, during some routine monitoring, I noticed that the Report Server was performing some strange queries that I knew I didn't write. Then it occurred to me that someone was trying to hack into the system. It was a small company so it was easy to find the culprit and remind them that hacking is inappropriate.

After handling the personnel end of the problem, I changed the string parameter to a list of available values — and that was the end of the injection attacks.

Mitigating HTML Injection Attacks in a Published Report or Document

In Reporting Services, reports and resources are processed under the security identity of the user who is actually running the report. If the report contains expressions, script, custom report items, or custom assemblies, the code runs under that user's credentials. If the resource happens to be an HTML document that contains script, the script will run when the user opens the document on the Report server. The ability to run script or code within a report is a powerful feature but comes with an increasing level of risk: if the code is malicious, the Report Server and the user running the report are vulnerable to attack.

Also, if a report parameter is not tied to a query parameter and the parameter values are included in the report, it is possible for a report user to type expression syntax or a URL into the parameter value and render the report to Excel or HTML. If another user then views the report and clicks the rendered parameter contents, the user may inadvertently execute the malicious script or link.

You can mitigate the risk of running malicious script by taking some precautions. First, be selective about who can publish content to a Report Server. Ideally this should be limited to a small number of trusted users. Also, make sure that all publishers are trained not to publish reports and resources coming from unknown or untrusted sources.

The Bottom Line

Explore the various types of authentication. Authentication is the process of making sure that a user is who they claim to be. Authentication is provided by a combination of `ASP.NET`, Internet Information Services (IIS), and Windows security.

Master It What types of authentication are available through IIS, and which ones can you use with Reporting Services?

Learn how to set up SSL. Secure Sockets Layer (SSL) provides an extra layer of security by encrypting all data that is transferred between the client and the server. It does this using a key that you can obtain from Windows Certificate Services or, better yet, from a third party such as VeriSign.

> **Master It** You need to make sure that sensitive report data and API calls cannot be opened by malicious users while they are being transferred over the network. How can you configure the Report Server to use SSL to protect all your data?

Understand how to secure folders, models, and My Reports. Once you have secured the server, you must secure the contents of the server. Reporting Services will ensure that users who are trying to access objects have permission to do so, a process called authorization. Reporting Services cannot do this unless you configure security for the objects you want authorization for. To start, you should secure your folders, models, and My Reports.

> **Master It** You have created a model for the Accounting group called AcctgModel. Most of the data in the model is available to all the employees in the Accounting group, but there is one entity, named EmployeeData, that should only be available to the Accounting management staff. How can you secure the model data?

Learn how to mitigate injection attacks. Injection attacks are the number-one security threat against databases. Using an injection attack, malicious users can insert their own SQL query into a text box in your application and gain access to data that they were not meant to see. Malicious users can also inject harmful scripts into web pages to accomplish similar goals.

> **Master It** You are creating a report that uses parameters to display sensitive employee data. What should you keep in mind in order to mitigate the chance of an injection attack?

Chapter 15

Data Sources and Data Sets

Reports are based on data, and to access that data, you need to create data sources and data sets. A report definition includes three important components:

- A data source definition, which includes the location of the source of data, the connection information, and credentials to use when connecting to the data source
- A data set, which includes the data source, the query command that specifies which data to retrieve, the collection of fields to retrieve, data options about how to interpret the data to retrieve, parameters to use in the query, and filters to use to limit the data retrieved
- Report parameters, which are automatically created from data set query parameters

You can use many different types of data sources, including relational databases, Analysis Services cubes, web services, XML data, and other document types. Four tools are available for managing report data sources:

- Report Designer
- Report Manager
- Management Studio
- Model Designer

When designing a report, you will need to make several decisions about the data set that the report is based on. Some of those are:

- Where does the data come from?
- How is it accessed?
- Do you need all of the data from the underlying data source, or just a subset?

In this chapter you will learn how to:

- Define data sources
- Define data sets

Defining Report Data Sources

One of the first things you need to do when you create a report is create a data source definition. The data source definition provides all the pertinent details on how to connect to the underlying data source, including the data source type, the connection string, and the credentials for connecting to the data source.

Data sources can be report specific or shared. Report-specific data sources are embedded right into the report definition and can only be used by the report that contains the information.

Shared data sources are separate objects that can be created and managed apart from the reports that reference them. Multiple reports can reference a single shared data source, which can simplify administration. Consider this: Suppose you use report-specific data sources in several reports that all connect to the same data source. If you move the data source to a new server, then you need to modify and republish every report that points to the data source. If you use a shared data source, you only need to modify the data source object and all of the reports that use it will be affected. Any report or data-driven subscription can use a shared data source, and report models require shared data sources.

The Reporting Services data processing extensions that use the data source definitions support a number of data types, such as relational databases, Analysis Services multidimensional cubes, report models, and XML-based data. You can also register and use the standard .NET data providers, but these do not provide the same functionality as the Reporting Services data processing extensions. The data returned from these data sources is usually returned in a tabular rowset, much like what you see when you execute a `SELECT` query in Management Studio. The following are the data processing extensions that are installed by default:

Microsoft SQL Server Data Processing Extension This extension wraps and extends the .NET Framework Data Provider for SQL Server. It runs on x86, x64, and Itanium-based systems. You can use this extension to connect to SQL Server 2005, SQL Server 2000, and SQL Server 7.0.

Microsoft SQL Server Analysis Services Data Processing Extension This extension wraps and extends the .NET Framework Provider for Microsoft Analysis Services. It runs on x86, x64, and Itanium-based systems. This uses the `ADOMD.NET` object model to create queries using XML for Analysis (XMLA) version 1.1. The result is returned as a flattened rowset. You can only use this extension to connect to SQL Server Analysis Services 2005 cubes.

SQL Server 2000 Analysis Services Use this extension to connect to SQL Server 2000 Analysis Services cubes. To use this extension, you must install the XMLA SDK Version 1.1 on the server that houses Analysis Services (which you can download from `http://go.microsoft.com/fwlink/?linkid = 81745`) and you must use the HTTP protocol; TCP/IP is not supported. Query parameters are not supported with this extension, either.

OLE DB Data Processing Extensions Several processing extensions extend the .NET Framework Data Provider for OLE DB:

OLE DB for SQL Server This allows you to use OLE DB to access SQL Server 2005, SQL Server 2000, SQL Server 7.0, or SQL Server 6.5.

OLE DB for Analysis Services 9.0 This allows you to use OLE DB to access SQL Server 2005 Analysis Services cubes, but it is not recommended because it does not support server aggregates, automatic mapping of extended field properties, or query parameters. You should use the Microsoft SQL Server Analysis Services extension instead.

OLE DB for Analysis Services 8.0 This allows you to use OLE DB to access SQL Server 2000 Analysis Services cubes, but it is not recommended because it does not support server aggregates, automatic mapping of extended field properties, or query parameters. You should use the Microsoft SQL Server Analysis Services extension instead.

OLE DB for Oracle This allows you to access Oracle databases using OLE DB, but it does not support the BLOB, CLOB, NCLOB, BFILE, or UROWID data types. Position-dependent,

unnamed parameters are not supported, either. I recommend that you use the Oracle processing extension instead.

OLE DB Standard .NET Framework Data Provider This lets you use the standard .NET Framework data provider with OLE DB to access data.

ODBC Standard .NET Framework Data Provider This extension wraps and extends the ODBC .NET Framework data provider. It runs on x86, x64, and Itanium-based systems.

Oracle Data Processing Extension This extension wraps and extends the .NET Framework Data Provider for Oracle and the `System.Data.OracleClient` classes. To use this provider you must install the Oracle client software version 8.1.7 or later. This extension supports named parameters, and Oracle version 9 supports multivalued extensions.

XML Data Processing Extension Use this extension to access XML data from an XML document, a web service, or a web-based application that can be accessed via a URL.

Data Provider for mySAP Business Suite This extension allows you to connect to mySAP Business Suite based on SAP R/C 4.6c or higher. You must download and install the provider first, which you can locate at `http://go.microsoft.com/fwlink/?linkid=48398`.

SAP NetWeaver Business Intelligence Data Processing Extension If you have installed Reporting Services Service Pack 1 or later, you have this extension, which allows you to connect to a SAP NetWeaver Business Intelligence data source.

Hyperion Essbase Intelligence Data Processing Extension If you have installed Reporting Services Service Pack 2 or later, you have this extension, which allows you to connect to a Hyperion Essbase data source.

Report Model Data Sources This extension allows you to connect to report models generated from SQL Server databases, Oracle 9.2.0.3 or later databases, and SQL Server 2005 Analysis Services multidimensional cubes. To use a model as a data source, it must be published to the server or a SharePoint library.

Table 15.1 shows some common connection strings used to connect to many of these data sources.

Creating a Shared Data Source

Some data sources are specific to the report that they are created for, which means that no other report can use them. Other data sources can be shared between multiple reports. When a shared data source is modified, the changes are reflected in every report that uses that data source. This approach cuts down on administrative overhead if you have multiple reports that use the same data source. Shared data sources consist of the following components:

- A name that identifies the item within the Report server folder hierarchy
- A description that provides information about the data source connection
- A connection type that identifies the data processing extension used with the data source
- The connection string used to log into the database
- Credential information that identifies how credentials are obtained, what values are to be used, and how they are to be used after the database connection is made

TABLE 15.1: Example Connection Strings

DATA SOURCE	EXAMPLE	NOTES
Microsoft SQL Server	`data source = "(local)";initial catalog = AdventureWorks`	Set data source type to SQL Server.
Microsoft SQL Server Analysis Services	`data source = localhost;initial catalog = Adventure Works DW`	Set data source type to SQL Server Analysis Services.
Report model data source	`Server = http://myreportservername/reportserver; datasource = /models/Adventure Works`	Specify the Report Server or document library URL and the path to the published model in the Report Server folder or document library folder namespace.
SQL Server 2000 Analysis Services server	`provider = MSOLAP.2;data source = <remote server name>;initial catalog = FoodMart 2000`	Set the data source type to OLE DB Provider for OLAP Services 8.0. Set the ConnectTo property to 8.0 on the Advanced tab of the Connection Properties dialog box to achieve a faster connection.
Oracle server	`data source = myserver`	Set the data source type to Oracle. The Oracle client tools version 8.1.7 or higher must be installed on the Report Designer computer and on the Report Server.
SAP NetWeaver BI data source	`DataSource = http://mySAPNetWeaver BIServer:8000/sap/bw/xml/soap/xmla`	Set the data source type to SAP NetWeaver BI.
Hyperion Essbase data source	`Data Source = http://localhost:13080/aps/XMLA; Initial Catalog = Sample`	Set the data source type to Hyperion Essbase.
XML data source	`data source = http://adventure-works.com/results.aspx`	Set the data source type to XML. The connection string is a URL to the XML document.

You may have noticed that the query is not listed here. That is because the query is always stored with the report definition.

You must also specify connection security settings for each data source. You can choose among five options:

Prompted Credentials Use this only for reports that run on demand. Prompted credentials can be a Windows account or a database login. To use Windows Authentication, you must select Use As Windows Credentials When Connecting To The Data Source. Otherwise, the Report Server passes credentials to the database server that hosts the external

data source. If the database server is not configured to use the login you provide, the connection will fail. This approach is recommended for reports that contain confidential data.

Windows Integrated Security The Windows Integrated Security option allows for the user to not be prompted to type a username or password and is recommended if Kerberos is enabled. If Kerberos is not enabled, you should use this approach if all the servers that you want to access are located on the same computer.

Stored Credentials You create stored credentials in reversible encryption in the Report Server database. You can then specify one set of stored credentials for each data source used in a report. The credentials you provide retrieve the same data for every user who runs the report.

The username and password that you specify can be a Windows username or a database login. If you specify Windows credentials, the Report Server asks Windows for authentication; otherwise, the credentials are passed to the database server for authentication.

Stored credentials are recommended as part of a strategy for accessing remote database servers. They are also required for supporting subscriptions, scheduling report history generation, and refreshing report snapshots.

Using Impersonation with Stored Credentials You can also use credentials to impersonate the identity of another user. For SQL Server databases, using the impersonation options sets the SETUSER function.

NOTE

Do not use impersonation for reports that may be used in scheduled operations, such as subscriptions, report history generation, or report execution snapshot refresh.

No Credentials You can configure a data source connection to use no credentials, though this is not generally recommended. If you do need to use this option, you must configure an unattended execution account for it to work properly because the Report Server does not connect to remote data sources using its own credentials. The circumstances where you might use this option are:

- The remote data source does not require credentials.
- The credentials are passed in the connection string (recommended only for secure connections).
- The report is a subreport that uses the credentials of the parent report.

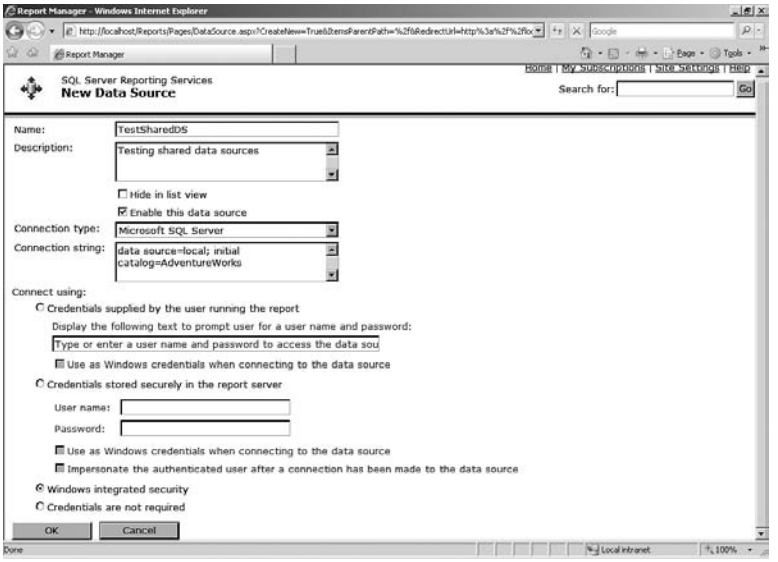
Let's create a shared data source here:

1. Open Report Manager and, on the Home page, click New Data Source.
2. Type **TestSharedDS** in the Name text box. A name must contain at least one character and none of these special characters: ; ? : @ & = + , $ / * < > — '' /.
3. Enter **Testing shared data sources** in the Description text box. This description will appear on the Contents page in Report Manager.
4. Select Microsoft SQL Server from the Connection Type list.

5. In the Connection String text box, enter this connection string to connect to the Adventure-Works database:

```
data source=local; initial catalog=AdventureWorks
```

6. Select the Windows Integrated Security option under Connect Using.



7. Click OK.

Real World Scenario

CASE STUDY: CHOCOLATY GOODNESS CANDIES

At CGC, we had a number of employees who requested data in different formats from the same data source. Some people wanted to see different fields, some wanted calculated fields, some wanted tabular formats, and others wanted matrix formats. The list of differences was endless, and the requests were so varied that we could not combine them into a single report — which meant we had to create multiple reports to meet these needs.

I realized right away that creating multiple reports with report-specific data sources was going to cause problems down the road because we were already planning on migrating the data source to a new, more powerful SQL Server machine (the Report Server would not move). When we implemented the new server, we would need to republish all the reports to compensate.

To avoid that problem, we created one shared data source for our SQL Server system and another shared data source for our Oracle server. When we migrated to the new SQL Server, all we had to do was modify the shared data source, and all the reports automatically pointed to the new server.

Using Data Source Expressions

Many large firms have multiple SQL Server systems that are geographically dispersed. You need to have your users connect to the server that is closest to them, but you will not know which database server that is when they run your report. That is where data source expressions come in handy.

A data source expression is just a dynamic connection string. Instead of inserting a server name, you insert a parameter in its place. When the user runs a report that uses the data source expression, they will be prompted for the server name. A data source expression that connects to the AdventureWorks database would look like this:

```
="data source=" & Parameters!ServerName.Value & ";initial catalog=AdventureWorks
```

Keep the following points in mind if you decide to use data source expressions:

- You cannot execute a query that uses a data source expression in Report Designer; the field list in the Datasets window and the Parameters list will not update automatically. So, you need to use a static connection string when you design the report.
- Do not use a shared data source when you define the data source connection because you cannot use a data source expression in a shared data source. You must define a report-specific data source for the report.
- Specify credentials separately from the connection string. You can use stored credentials, prompted credentials, or integrated security.
- Add a report parameter to specify a data source. For parameter values, you can provide a static list of available values. You can also create a table that holds a list of available servers and define a query that retrieves a list of data sources from that table at runtime.
- All report design begins with the database schema, so ensure that all the data sources in the list have the same database schema. If there is a mismatch between the schema used to define the report and the actual schema used by the report at runtime, the report might not run.
- Be sure to replace the static connection string that you use during development with a data source expression before publishing the report to production.

Defining Report Data Sets

After you have created a data source for your report, you can create a data set. You create one data set for each data source in your report. A data set contains several settings:

- The fields you will use from the data source
- Any calculated fields you may create
- A pointer to the data source
- Any parameters you may create

- Data options, such as:
  - Case sensitivity
  - Collation settings
  - Kana sensitivity
  - Width sensitivity
  - Accent sensitivity

All of the data sets you create for a report are displayed in the Report Datasets window, where you can add fields or edit existing fields. Once you have finished editing the data set, you can drag and drop fields from the Report Datasets window onto the report design surface to create a report. The data set is not populated with data from the data source until the report is rendered.

The way you define the data set is dependent on the data source. For example, if you are connecting to a SQL Server database, you would use Transact-SQL to define the data set. If you are connecting to a SQL Server Analysis Services cube, you would use a multidimensional or data-mining query. If you are connecting to a report model, you could use a report model metadata query. There are as many options as there are data sources. In this section, you will learn how to create data sets based on these data sources.

SQL Server Relational Database Data Sets

When you create a data set for a relational database, you have two query designers to work with: the generic designer and the graphical designer. The generic designer lets you enter text directly, which is handy for building complex queries. The graphical designer lets you browse the metadata to define a data set, which is useful when you do not know the schema of the underlying data source.

You can create data sets for SQL Server relational databases using the Microsoft SQL Server data processing extension or the OLE DB for SQL Server data processing extension. Depending on the processing extension you choose, three command types are available for defining data sets:

`Text` This command type allows you to enter a Transact-SQL query to define your data set. If your query contains parameters, Report Designer will automatically generate report parameters for the query. A query with parameters would look something like this:

```
SELECT FirstName, LastName
FROM HumanResources.Employee E INNER JOIN
Person.Contact C ON E.ContactID=C.ContactID
WHERE EmployeeID = @EmpID
```

`StoredProcedure` You can use the `StoredProcedure` command type to define a data set based on a stored procedure. The stored procedure should only return a single set of data; if it returns multiple sets, then only the first set is used. If you are using a stored procedure with a default value, you can access that value by using the `DEFAULT` keyword.

`TableDirect` This command type is only available when you use the OLE DB for SQL Server data processing extension. This command type accepts a table name as a parameter and returns every column and row in the table to define the data set.

To create a data set based on the Microsoft SQL Server data processing extension:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.
4. In the Data Source dialog box, from the Type drop-down list select Microsoft SQL Server.
5. Specify a connection string that works with your SQL Server data source. This example would connect you to the AdventureWorks database on the local server:

```
Data Source=localhost;Initial Catalog=AdventureWorks;
```

6. Set the database credentials in the Data Source dialog box on the Credentials tab.
7. Click OK.
8. Enter a query in the Query String text box to populate the data set.
9. Click OK.

Analysis Services Data Sets

Several data processing extensions let you retrieve data from SQL Server Analysis Services to define a data set. Using these data processing extensions, you can retrieve:

- Dimensions
- Hierarchies
- Levels
- Key Performance Indicators (KPIs)
- Measures
- Attributes

You can retrieve this data using either Multidimensional Expression (MDX) queries or Data Mining Prediction (DMX) queries. Each query type provides a different query designer.

The MDX Designer

You can use the MDX designer in either design or query mode. Design mode provides a graphical method for designing your data set. You can drag and drop elements from the cube onto the design surface to create an MDX query. As shown in Figure 15.1, the query designer has four panes in design mode:

Metadata Pane This pane lets you view and work with the metadata from the underlying cube. You can drag objects from this pane to the Data pane to create columns for the result set. You can drag objects from the Metadata pane to the Filter pane to create filters

for the result set. You can also drag these objects from the Metadata pane to the Data and Filter panes:

- Dimensions
- Levels
- Members
- Measures
- Hierarchies and attributes
- KPIs
- Named sets and members (to Filter pane only)

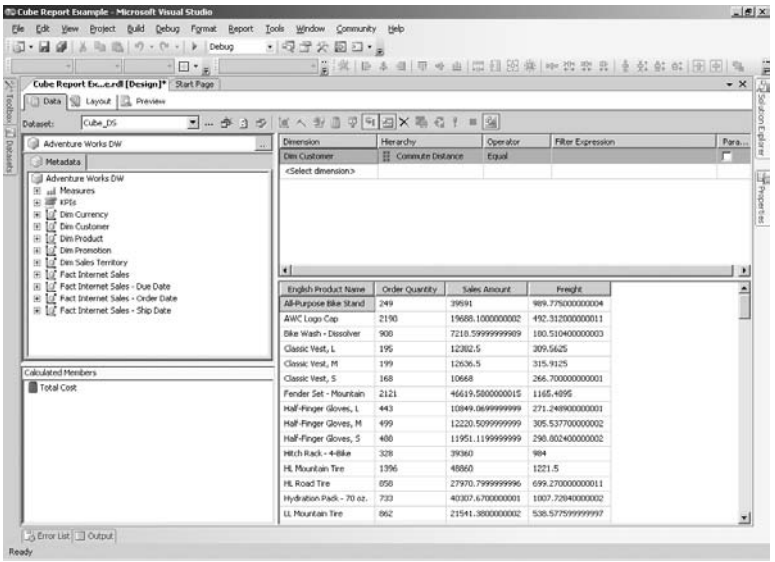


FIGURE 15.1
The MDX Designer has four panes in design mode.

Calculated Members Pane A *calculated member* is a custom member that does not exist in the underlying data source — you must create it. For example, if you have a member that holds the item cost for an order and another member that holds the shipping cost, you could create a calculated member to combine the two and show total order cost. The Calculated Members pane allows you to create these calculated columns and drag them to the design surface to add them to the MDX query.

Filter Pane In this pane you can create filters to limit the amount of data returned in the result set. You can select the dimension and hierarchy on which to filter and set the operator and filter expression to apply.

Data Pane In the Data pane, you interactively build the MDX query that you want to execute to return the result set.

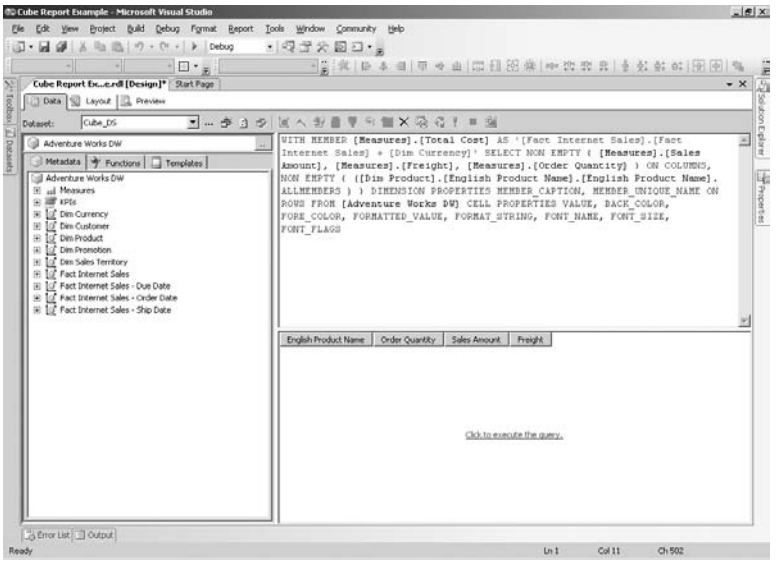


FIGURE 15.2
The MDX Designer has three panes in query mode.

You can switch the designer to query mode to enter MDX queries manually by clicking the Design Mode button on the toolbar. There are only three panes to work with in query mode, as shown in Figure 15.2:

Metadata Pane This pane lets you view and work with the metadata from the underlying cube. You can drag these objects from the Metadata pane to the Query pane to create columns for the result set:

- Levels
- Members
- Hierarchies and attributes
- KPIs

Query Pane You can drag objects on to the Query pane to define the data set, or you can manually enter a query.

Result Pane The Result pane shows you the data that will be returned from the query you have created.

Follow these steps to create an MDX-based data set:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab, select New Dataset from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.

4. In the Data Source dialog box, from the Type drop-down list select Microsoft SQL Server Analysis Services.
5. Specify a connection string that works with your Analysis Services data source. This example would connect you to the Adventure Works DW sample data warehouse:

```
Data Source=(local);Initial Catalog="Adventure Works DW"
```

6. Set the database credentials in the Data Source dialog box on the Credentials tab.
7. Click OK.

The DMX Designer

The DMX designer is displayed by default when you create a data set based on Analysis Services, so you need to click the Command Type DMX button on the toolbar to access the DMX designer (it has an icon that looks like a pickax).

Using the DMX designer, you can create Data Mining Prediction queries to retrieve data from a data-mining model that has been previously created. The DMX designer opens in design mode, which allows you to create a DMX query graphically by dragging and dropping elements onto the design surface. You can switch to query mode to enter a DMX query manually, and you can switch to result mode to see the data returned by the query in a grid format. Figure 15.3 shows the DMX designer in design mode.

Here are the steps to create a DMX-based data set:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.
4. In the Data Source dialog box, from the Type drop-down list select Microsoft SQL Server Analysis Services.

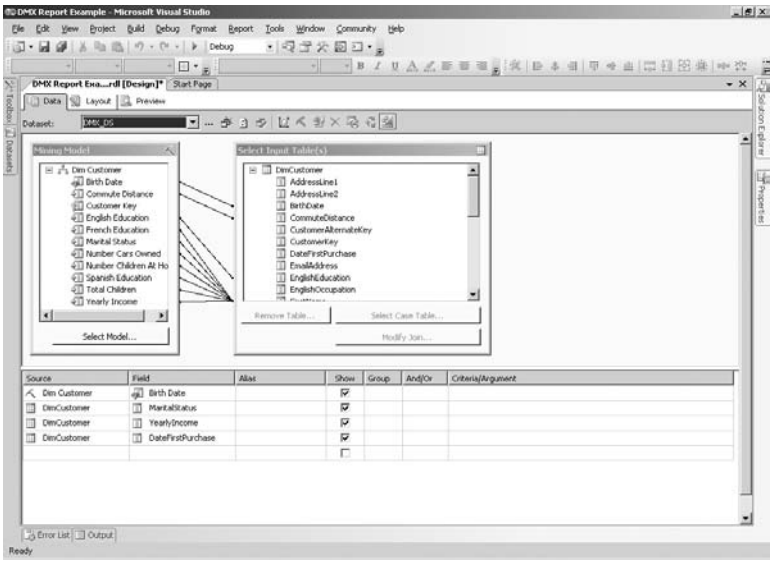


FIGURE 15.3
The DMX Designer opens in query mode by default.

5. Specify a connection string that works with your Analysis Services data source. This example would connect you to the Adventure Works DW sample data warehouse:

```
Data Source=(local);Initial Catalog="Adventure Works DW"
```

6. Set the database credentials in the Data Source dialog box on the Credentials tab.
7. Click OK.

SQL Server Integration Services Data Sets

You can create a data set based on data returned from a SQL Server Integration Services (SSIS) package. First, you must configure an unattended execution account and configure the server to use the SSIS data processing extension. Here are the steps to enable the SSIS data processing extension for the Report Designer:

1. Install Integration Services on the same computer on which you installed Reporting Services.
2. Using Windows Explorer, locate *<drive>*:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies.
3. Using a text editor, open the RSReportDesigner.config file.
4. Remove the comment marks (delete <!-- and --> characters from the beginning and end of the line) for this entry in the <Data> section:

```
<!--<ExtensionName="SSIS"➢
Type="Microsoft.SqlServer.Dts.DtsClient.➢
DtsConnection,Microsoft.SqlServer.Dts.DtsClient,➢
Version=9.0.242.0,Culture=neutral,➢
PublicKeyToken=89845dcd8080cc91"/> -->
```

5. In the <Designer> section, remove the comments from this entry:

```
<!--<ExtensionName="SSIS"➢
Type="Microsoft.ReportingServices.QueryDesigners.➢
GenericQueryDesigner,Microsoft.ReportingServices.➢
QueryDesigners"/> -->
```

6. Save the file.

Then follow these steps to configure the extension for the Report Server:

1. Open the RSReportServer.config file, located by default at *<drive>*:\Program Files\Microsoft SQL Server\MSSQL.*n*\Reporting Services\ReportServer.
2. Remove the comment marks (delete <!-- and --> characters from the beginning and end of the line) for this entry:

```
<!--<ExtensionName="SSIS"➢
Type="Microsoft.SqlServer.Dts.DtsClient.DtsConnection,➢
Microsoft.SqlServer.Dts.DtsClient,Version=9.0.242.0,➢
Culture=neutral, PublicKeyToken=89845dcd8080cc91"/> -->
```

3. Save the file.

After the extension is enabled, you can use SSIS packages as data sets. You cannot use all SSIS packages, though; you must use a DataReader destination that provides an in-memory destination. Here are the steps to create a viable package:

1. In Integration Services, create a package that has at least one data flow task that fills one or more DataReader destinations. The destination object must include the input columns that you want to use in your report.
2. Configure a connection to an external data source. If the package connects to a SQL Server relational data source, use SQL Server Authentication with a stored password.
3. Run the package to verify it returns the data you expect.
4. Save the package to a folder on the filesystem or SQL Server. Saving the package to the SSIS Package Store is not supported.
5. Note the name of the DataReader destination that contains the result set you want to use in the report. You will specify this name later as the report query.

Here are the steps to create an SSIS data set:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.
4. In the Data Source dialog box, from the Type drop-down list select Microsoft SQL Server Integration Services.
5. The connection string must be the fully qualified path to the package. This example would read a package stored on the filesystem:

```
-f c:\packages\ReportPackage.dtsx
```

6. Use No Credentials or Windows Credentials in the Data Source dialog box on the Credentials tab. Database Credentials are not supported with SSIS packages.
7. Click OK.

XML Data Sets

Using the XML data processing extension, you can retrieve data from XML documents stored on the filesystem or web services, or you can embed your own XML directly into the query. You cannot retrieve data from an XML document stored in a SQL Server database.

Here are the steps to create an XML data set:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.

4. In the Data Source dialog box, from the Type drop-down list select XML.
5. The connection string must be a valid URL to the XML data or an empty value. This example would connect to a web service:

```
http://www.CorpDomain.com/ReportSource.aspx
```

   This example would connect to a file in the XMLSource directory:

```
http://localhost/XMLSource/ReportSource.xml
```

6. Use No Credentials or Windows Credentials in the Data Source dialog box on the Credentials tab. Database Credentials are not supported with XML data.
7. Click OK.

Oracle Relational Database Data Sets

The Oracle data provider extensions work in a similar fashion to the SQL Server data processing extensions. They allow you to connect to an Oracle database using a text query written in PL/SQL or by executing a stored procedure on the Oracle database.

Here are the steps to create a data set based on the .NET Managed Provider for Oracle data processing extension:

1. Create a new report in Business Intelligence Design Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.
4. In the Data Source dialog box, from the Type drop-down list select Oracle.
5. Enter a connection string in the Connection String text box. The server name must match the name in the `tnsnames.ora` file. This example would connect you to an Oracle server named CorpOra:

```
Data Source="CorpOra";Unicode="true"
```

6. Set the username and password required to connect to the database on the Connections tab.
7. Click OK.

Here are the steps to create a data set based on the OLE DB for Oracle data processing extension:

1. Create a new report in Business Intelligence Design Studio.
2. On the Data tab, select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data Source drop-down list, or select a shared data source if one is available.

4. In the Data Source dialog box, from the Type drop-down list select OLE DB.
5. Click Edit to open the Connection Properties dialog box.
6. Choose the Microsoft OLE DB Provider for Oracle from the OLE DB Provider drop-down list.
7. Enter a connection string in the Connection String text box. The server name must match the name in the `tnsnames.ora` file. This example would connect you to an Oracle server named CorpOra:

```
Provider="MSDAORA.1"; Data Source="CorpOra"
```

8. Set the username and password required to connect to the database on the Connections tab.
9. Click OK.

SAP Relational Database and NetWeaver BI System Data Sets

You can create a data set using data from the mySAP business suite, but you need to install and configure the provider first because it is not installed by default. After downloading and installing the data processing extension, follow these steps to configure it for use.

First, you have to add a code group to the `RSPreviewPolicy.config` file:

1. Open the `RSPreviewPolicy.config` file, located by default at `C:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies`.
2. Add the following code group:

```
<CodeGroupclass="UnionCodeGroup"version="1"
PermissionSetName="FullTrust"
Name="Microsoft\_Generic\_Signing"Description="This
codegroupgrantscodesignedwiththegenericMicrosoft
keyfulltrust.UsedbytheMicrosoft.NetData
ProviderformySAPBusinessSuite.">
<IMembershipCondition
class="StrongNameMembershipCondition"version="1"
PublicKeyBlob="00240000048000009400000006020000002400005 2
534131000400000100010 0b5fc90e7027f67871e773a8fde8938c81dd402
ba65b9201d60593e96c492651e889cc13f1415ebb53fac1131ae0bd333c5
ee6021672d9718ea31a8aebd0da0072f25d87dba6fc90ffd598ed4da35e4
4c398c454307e8e33b8426143daec9f596836f97c8f74750e5975c64e218
9f45def46b2a2b1247adc3652bf5c308055da9"/>
</CodeGroup>
```

3. Save the file.

Next you need to make some modifications to the `RSReportDesigner.config` file:

1. Using Windows Explorer, locate *X*:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies.
2. Using a text editor, open the `RSReportDesigner.config` file.

3. Remove the comment marks (delete < !-- and -- > characters from the beginning and end of the line) for this entry in the < Data > section:

```
<!--ExtensionName="SAP"➢
Type="Microsoft.Adapter.SAP.SAPConnection,Microsoft.➢
Adapter.SAP.SAPProvider,Version=1.0.0.0,Culture=neutral,➢
PublicKeyToken=31bf3856ad364e35"/ -->
```

4. In the Designer section, remove the comments from this entry:

```
<!--ExtensionName="SAP"➢
Type="Microsoft.ReportingServices.QueryDesigners.➢
GenericQueryDesigner,Microsoft.ReportingServices.➢
QueryDesigners"/ -->
```

5. Save the file.

Then you have to add a code group to the `RSSrvPolicy.config` file:

1. Open the `RSSrvPolicy.config` file, located by default at *X*:\Program Files\Microsoft SQL Server\MSSQL.*n*\Reporting Services\ReportServer.

2. Add the following code group:

```
<CodeGroupclass="UnionCodeGroup"version="1"
PermissionSetName="FullTrust"
Name="Microsoft\_Generic\_Signing"Description="This
codegroupgrantscodesignedwiththegenericMicrosoft
keyfulltrust.UsedbytheMicrosoft.NetData
ProviderformySAPBusinessSuite.">
<IMembershipCondition
class="StrongNameMembershipCondition"version="1"
PublicKeyBlob="00240000048000009400000006020000002400005200
5341310004000001000100b5fc90e7027f67871e773a8fde8938c81dd402
ba65b9201d60593e96c492651e889cc13f1415ebb53fac1131ae0bd333c5
ee6021672d9718ea31a8aebd0da0072f25d87dba6fc90ffd598ed4da35e4
4c398c454307e8e33b8426143daec9f596836f97c8f74750e5975c64e218
9f45def46b2a2b1247adc3652bf5c308055da9"/>
</CodeGroup>
```

3. Save the file.

Finally, you need to make some modifications to the `RSReportServer.config` file:

1. Open the `RSReportServer.config` file, located by default at *X*:\Program Files\Microsoft SQL Server\MSSQL.*n*\Reporting Services\ReportServer.

2. In the <Data> section, remove the comments from this entry:

```
<ExtensionName="SAP"
Type="Microsoft.Adapter.SAP.SAPConnection,Microsoft.
Adapter.SAP.SAPProvider,Version=1.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"/>
```

3. Save the file.

Once you have configured the system to work with the SAP systems, you can create a data set that returns relational data from the mySAP Business Suite or multidimensional data from the NetWeaver BI System. Follow these steps to create a multidimensional data set:

1. On the Data tab, from the Dataset drop-down list, select <New Dataset> . In the Dataset dialog box, from the Data Source drop-down list select New Data Source. In the Data Source dialog box, from the Type drop-down list select SAP NetWeaver BI.
2. Specify a connection string that works with your SAP NetWeaver Business Intelligence data source. Contact your database administrator for connection information. The following connection string example specifies an SAP NetWeaver Business Intelligence data source on a server using port 8000 and XML for Analysis Services (XMLA) over the Internet using SOAP:

```
DataSource=http://SAPSvr:8000/sap/bw/xml/soap/xmla
```

3. Set the database credentials in the Data Source dialog box on the Credentials tab.

Hyperion Essbase Data Sets

You can create a data set based on Hyperion Essbase for your report. The Hyperion Essbase data processing extension provides a graphical query designer that runs in design and query modes.

In design mode, you can drag and drop elements onto the design surface to create a multidimensional query. In query mode, you can enter a query manually.

Follow these steps to create a data set for Hyperion Essbase:

1. On the Data tab, from the Dataset drop-down list, select New Dataset.
2. In the Dataset dialog box, from the Data Source drop-down list, select New Data Source. In the Data Source dialog box, from the Type drop-down list, select Hyperion Essbase.
3. Specify a connection string that works with your Hyperion Essbase data source. Contact your database administrator for connection information. The following connection string example specifies a Hyperion Essbase data source on a server using port 13080 and XML for Analysis Services (XMLA) over the Internet using SOAP, connecting to a sample catalog:

```
DataSource=http://Essbase:13080/aps/XMLA;
Initial Catalog=Sample
```

4. Set the database credentials in the Data Source dialog box on the Credentials tab.

Report Model Data Sets

As we discussed in Chapter 8, a report model is a layer of metadata that describes the database and its relationships in user-friendly terms. Report models can be based on:

- SQL Server 2000 databases
- SQL Server 2005 databases

- SQL Server 2005 Analysis Services Cubes
- Oracle 9.2.0.3 or later databases

The Report Model data processing extension provides a graphical query designer that allows you to drag and drop elements onto the design area to create a data set. To create a data set based on a report model, you first need to create the report model and publish it to the server successfully.

Follow these steps to create a data set based on a report model:

1. On the Data tab, from the Dataset drop-down list select New Dataset.
2. In the Dataset dialog box, from the Data Source drop-down list, select New Data Source. In the Data Source dialog box, from the Type drop-down list, select Report Server Model.
3. Specify a connection string to the model using this format:

```
Server=http://server/reportserver;datasource=/modelpath/modelname
```

4. Set the database credentials in the Data Source dialog box on the Credentials tab.

The Bottom Line

Define data sources. Data source definitions contain all the information needed to connect to an underlying data source, including the data source type, the connection string, and the credentials for connecting to the data source. Data sources can be report specific or shared among multiple reports.

Master It Your developers need to create several reports that all use data from the same SQL Server database. You realize that it is best to create a shared data source for all of these reports to use. How would you do create the shared data source?

Define data sets. You can create one data set for each data source in your report. A data set contains such data as which fields you will use from the data source, calculated fields you may create, a pointer to the data source, parameters, and data options like case sensitivity and collation settings.

Master It You need to be able to use SSIS packages to create data sets for some of your reports. What must you do to enable this functionality?

Master It You are creating a new report that uses the FirstName, LastName, and SSN columns from the Employee table in your HumanResources database. The database is on a SQL Server 2005 system. How would you create the data set?

Chapter 16

Performance Tuning

People do not want to wait for important data. When they need data to make business decisions, they want it quickly. So you should make sure that your Report Server is running as fast as possible at all times. The only way to be certain of that is to monitor the server performance.

You can find performance data in three places:

- Task Manager
- Event Viewer
- Performance Console

Task Manager provides information about the processes that are running on your computer. You can use Task Manager to get a high-level overview of the system resources that are in use and assess the activity of running processes.

Event Viewer contains information on all the activities that have occurred on your system. If you believe that your server is not running as fast as it should, you can check Event Viewer for a record of errors and user activity to help assess the issue.

Task Manager and Event Viewer are a basic starting point. Performance Monitor, on the other hand, is specifically designed to help you monitor system performance.

The Reporting Services Windows service and web service both have associated performance counters that you can view in the Performance Console. This chapter will look at the available counters for each service and show you how to use the Performance Console to view them.

While you are monitoring Reporting Services, you may see the need to set the amount of memory that the system uses, so we will look at how to set the available memory for Reporting Services.

Many of the changes that you will make to configure Reporting Services for better performance are done through the Reporting Services configuration files, so this chapter examines each of those in depth.

In this chapter you will learn how to:

- Use performance counters
- Specify settings in each of the configuration files
- Monitor and configure Reporting Services

Using the Performance Console

Reporting Services cannot function properly if it does not have available system resources such as memory, processor power, fast disks, and a reliable network subsystem. If these systems do not work together, the overall system will not function properly. For example, if the memory is

being overused, the disk subsystem slows down, because the memory has to write to the pagefile (which is on the disk) far too often. To keep such things from happening, you need reports from the subsystems; you can do this using the Performance Console.

NOTE

You will find a brief overview of Performance Console in Chapter 7, which gave just enough detail to help you determine if your server could handle a Reporting Services installation. This section provides the level of detail you need to monitor performance.

The Performance Console comes with Windows and is located in the Administrative Tools folder on the Start menu (it is labeled Performance). Four views are available for your use:

Graph This view displays a graph of system performance. As values change, the graph spikes or dips accordingly.

Report The Report view looks like what you might get on a piece of paper, except that the values here change with system use.

Alert With Alert view, you can tell Performance Console to warn you when something bad is looming on the horizon, perhaps when CPU use is almost — but not quite yet — too high. This type of warning gives you time to fix potential problems before they become actual problems.

Log This view is for record keeping. With Log view, you can monitor your system over a period and view the information later, as opposed to viewing it in real time (the default).

With each of these views, you monitor objects and counters. An object is a part of the system, such as the processor or the physical memory. A counter displays the statistical information about how much that object is being used. For example, the % Processor Time counter under the Processor object tells you how much time your processor spends working. Table 16.1 lists common counters and their recommended values.

NOTE

To see the Network Segment: % Network Utilization, you must install the Network Monitor Agent; choose Control Panel ➢ Network and click the Services tab.

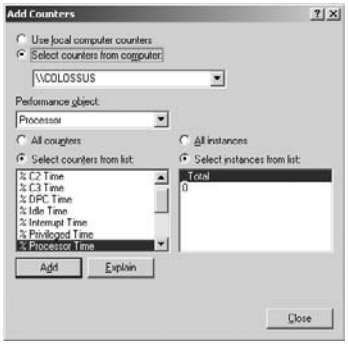
Now let's get some practice with Performance Console:

1. Log in to Windows as Administrator.
2. From the Start menu, select Programs ➢ Administrative Tools ➢ Performance. Notice that the graph is already populated with counters.

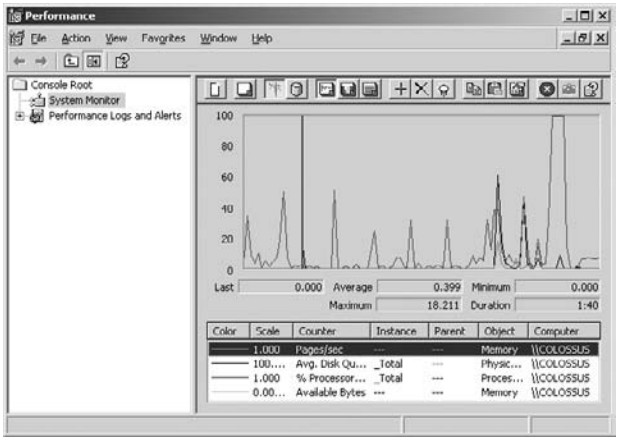
TABLE 16.1: Common Counters and Values in Performance Console

OBJECT	COUNTER	RECOMMENDED VALUE	USE
Processor	% Processor Time	Less than 75%	The amount of time the processor spends working.
Memory	Pages/Sec	Fewer than 5	The number of times per second that data had to be moved from RAM to disk, and vice versa.
Memory	Available Bytes	More than 4 MB	The amount of physical RAM available. This number should be low, because NT uses as much RAM as it can grab for file cache.
Memory	Committed Bytes	Less than physical RAM	The amount of RAM committed to use.
Disk	% Disk Time	Less than 50%	The amount of time that the disk is busy reading or writing.
Network Segment	% Network Utilization	Less than 30%	The amount of network bandwidth being used.

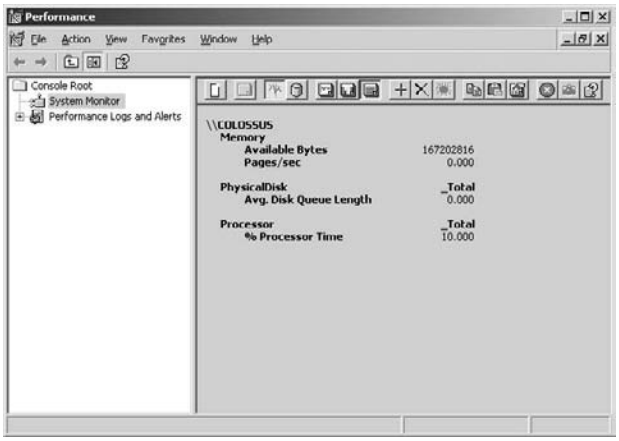
3. On the toolbar, click the Add button (it looks like a plus sign) to bring up the Add Counters dialog box.



4. In the Performance Object box, select Memory.
5. In the Counter box, select Available Bytes and click Add.
6. Click Close and notice the graph being created on the screen.



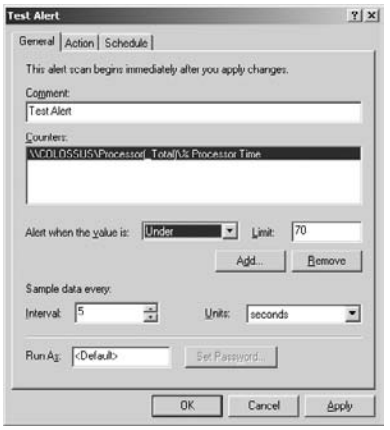
7. Press Ctrl+H and notice the current counter turns white. This makes the chart easy to read.
8. On the toolbar, click the View Report button (it looks like a sheet of paper) and notice how the same data appears in Report view.



9. In the left pane, expand Performance Logs and Alerts, right-click Alerts, and select New Alert Settings.
10. Enter **Test Alert** in the Name box and click OK.
11. In the Alert Settings dialog box, enter **Test Alert** in the Comment field.
12. Click Add.
13. Select the Processor object and the % Processor Time counter, click Add, and then click Close.

14. Select Under from the Alert When Value Is drop-down list, enter **70** for the Limit, and click OK. This will generate an alert if the processor is not busy 70 percent of the time. In the real world, this would be set to over 70 percent, thus warning you just before it becomes a serious problem.

15. Click OK to create the alert.



16. To view the alerts, open Event Viewer and look for them in the Application log.

17. Watch the alerts generated for a short time, and then select the alert and press the Delete key on the keyboard.

18. Exit Performance Console.

You can monitor SQL Server Reporting Services objects as well as Windows objects using Performance Console. The Windows service and web service both install their own performance counters that you can use to monitor Reporting Services performance. The process for monitoring Reporting Services is the same as it is with Windows — you just add different objects and counters. The counters to use for the web service are listed in Table 16.2 and the counters for the Windows service appear in Table 16.3.

TABLE 16.2: MSRS 2005 Web Service Object Counters

COUNTER	DESCRIPTION
Active Sessions	This gives the number of sessions generated by report subscriptions, even if they are not still active. Sessions are decremented after 10 minutes of inactivity.
Cache Hits/Sec	Number of requests per second for re-rendered reports, not requests for reports processed directly from the cache.

(CONTINUED)

TABLE 16.2: MSRS 2005 Web Service Object Counters *(CONTINUED)*

COUNTER	DESCRIPTION
Cache Misses/Sec	Number of requests per second that failed to return a report from cache. A high number of failures (as determined by your performance baseline) indicates that your server is running low on memory.
First Session Requests/Sec	Number of new user sessions that are started from the Report Server cache each second.
Memory Cache Hits/Sec	Number of times per second that reports are retrieved from the in-memory cache, which is a part of the cache that stores reports in CPU memory. When in-memory cache is used, the Report Server does not query SQL Server for cached content.
Memory Cache Misses/Sec	Number of times per second that reports could not be retrieved from the in-memory cache.
Next Session Requests/Sec	Number of requests per second for reports that are open in an existing session, such as reports that are rendered from a session snapshot.
Report Requests	Number of active reports that the Report Server is currently processing.
Reports Executed/Sec	Number of successful report executions per second. Combine this with Request/Sec to compare report execution to report requests that can be returned from cache.
Requests/Sec	Number of requests per second made to the Report Server. This tracks all requests, not just report requests.
Total Cache Hits	Total number of requests for reports from the cache after the service started. This counter is reset whenever ASP.NET stops the Report Server web service.
Total Cache Misses	Total number of times that a report could not be returned from the cache after the service started. This counter is reset whenever ASP.NET stops the Report Server web service. A high number indicates that you may not have enough memory.
Total Memory Cache Hits	Total number of cached reports returned from the in-memory cache after the service started. This counter is reset whenever ASP.NET stops the Report Server web service.
Total Memory Cache Misses	Total number of cache misses against the in-memory cache after the service started. This counter is reset whenever ASP.NET stops the Report Server web service.
Total Processing Failures	Total number of report processing failures that have occurred since the service started. This counter is reset whenever ASP.NET stops the Report Server web service.
Total Rejected Threads	Total number of data processing threads rejected for asynchronous processing, and subsequently handled as synchronous processes in the same thread.

TABLE 16.2: MSRS 2005 Web Service Object Counters *(CONTINUED)*

COUNTER	DESCRIPTION
Total Reports Executed	Total number of reports that ran successfully after the service started. This counter is reset whenever ASP.NET stops the Report Server web service.
Total Requests	Total number of all requests made to the Report Server after the service started whether or not they are successful. This counter is reset whenever ASP.NET stops the Report Server web service.

TABLE 16.3: MSRS 2005 Windows Service Object Counters

COUNTER	DESCRIPTION
Active Sessions	Number of active sessions stored in the Report Server database. This counter provides a cumulative count of all usable browser sessions generated from report subscriptions, whether or not they are still active. Sessions are decremented from the counter after 10 minutes of inactivity.
Cache Flushes/Sec	Number of cache flushes per second.
Cache Hits/Sec	Number of requests per second for re-rendered reports, not requests for reports processed directly from the cache.
Cache Misses/Sec	Number of requests per second that failed to return a report from cache. A high number of failures (as determined by your performance baseline) indicates that your server is running low on memory.
Delivers/Sec	Number of reports delivered per second from any delivery extension.
Events/Sec	Number of `SnapshotUpdated` and `TimedSubscription` events processed per second.
Memory Cache Hits/Sec	Number of times per second that reports are retrieved from the in-memory cache, which is a part of the cache that stores reports in CPU memory. When in-memory cache is used, the Report Server does not query SQL Server for cached content.
Memory Cache Misses/Sec	Number of times per second that reports cannot be retrieved from the in-memory cache.
Next Session Requests/Sec	Number of requests per second for reports that are open in an existing session such as reports that are rendered from a session snapshot.

(CONTINUED)

TABLE 16.3: MSRS 2005 Windows Service Object Counters *(CONTINUED)*

COUNTER	DESCRIPTION
Report Requests	Number of active reports that the Report Server is currently processing.
Reports Executed/Sec	Number of reports successfully generated per second.
Total App Domain Recycles	Total number of times the application domain was recycled after the Report Server Windows service started.
Total Cache Flushes	Total number of times the Report Server cache was updated after the service started. This counter resets when the application domain recycles.
Total Cache Hits	Total number of requests for reports processed directly from the cache after the Report Server Windows service started. This counter resets when the application domain recycles.
Total Cache Misses	Total number of times that a report could not be returned from cache after the Report Server Windows service started. This counter resets when the application domain recycles.
Total Deliveries	Total number of reports delivered by the Scheduling and Delivery Processor, for all delivery extensions. This counter resets when the application domain recycles.
Total Events	Total number of events after the Report Server Windows service started. This counter resets when the application domain recycles.
Total Memory Cache Hits	Total number of cached reports returned from the in-memory cache after the Report Server Windows service started. This counter resets when the application domain recycles.
Total Memory Cache Misses	Total number of cache misses against the in-memory cache after the service started. This counter resets when the application domain recycles.
Total Processing Failures	Total number of report processing failures that have occurred after the service started. This counter resets when the application domain recycles.
Total Rejected Threads	Total number of data processing threads rejected for asynchronous processing, and subsequently handled as a synchronous process in the same thread. This counter steadily increases under moderate to heavy load.
Total Reports Executed	Total number of reports run, whether or not they were successful.
Total Requests	Total number of reports that ran successfully after the service started. This counter resets when the application domain recycles.

Modifying Configuration Files

All of the settings that Reporting Services requires to run are stored in configuration files that are saved on the hard disk. Some of the values in these files are available for you to edit; others are for internal use only and should not be edited.

The configuration files all contain XML data, so you can use any XML editor or text editor to modify the files. Some of the settings are encrypted and you must use Reporting Services tools to modify them.

If you make changes to these files that are valid, the server incorporates your changes into the current session seamlessly. The results vary if the changes you make are not valid, but they are universally bad.

If you modify a setting in a configuration file and set it to an invalid value, the Report Server will try to use a default value for the setting if one is available. If there is no default value available, the server will fail to start next time you try to start it or the next time the application domain is recycled.

If you accidentally modify the XML so that it is no longer well formed, the server will fail to start the next time you try to start it or the next time the application domain is recycled. No default value or failsafe mechanism is available that can prevent this.

Here's the moral of the story: be careful when you edit the configuration files so that you do not bring your server to a grinding halt. That being said, let's look at the settings available in each of the configuration files.

ReportingServicesService

This configuration file contains settings that the Report Server uses to configure tracing, and it is saved in the `\Reporting Services\Report Server\Bin` directory.

As we discussed in Chapter 10, trace logs contain detailed information about Report Server operations that you can use for troubleshooting. There are four trace logs, but the `ReportServicesService.config` file only contains settings to configure two of them:

- The `ReportServerService_`*`timestamp`*`.log` file contains data on the Report Server Windows service worker threads.
- The `ReportServerService_main_`*`timestamp`*`.log` file contains data on the Report Server Windows service management threads.

You should only modify this file to change trace levels, rename the log files, or change the directory the log files are saved in. The settings for this file are listed in Table 16.4.

All of the configuration settings in this section are described in the .NET Framework product documentation. For more information, search for "Runtime Schema Settings" on the MSDN website or in the .NET Framework documentation.

RSReportDesigner

This file is used to store information about rendering and data processing extensions that are available to the Report Designer, and it is saved in the `\Program Files\Microsoft Visual Studio`

TABLE 16.4: *ReportingServicesService* Configuration File Settings

SETTING	DESCRIPTION
RStrace	Specifies namespaces used for errors and tracing.
DefaultTraceSwitch	Specifies the level of information that is reported to the ReportServerService trace log. Each level includes the information reported by all lower-numbered levels. Disabling tracing is not recommended. Valid values include: 0 = Disables tracing. 1 = Exceptions and restarts. 2 = Exceptions, restarts, and warnings. 3 = Exceptions, restarts, warnings, and status messages. 4 = Verbose mode.
FileName	Defines the portion of the filename before the last underscore.
FileSizeLimitMb	Defines the maximum size for the trace log. The default is 32.
KeepFilesForDays	Specifies how many days a trace log can remain on disk before it is deleted.
Prefix	Defines the portion of the filename after the last underscore.
TraceListeners	Specifies the target for the trace log output. Valid values are DebugWindow, File, and StdOut. DebugWindow and File are the defaults.
TraceFileMode	Do not modify this value; it is for internal use. This setting specifies whether the trace log should contain data for a 24-hour period.
Components	Defines which components are traced and at what level. You should not modify this value.
Runtime	Specifies configuration settings that support backward compatibility with the previous version. Runtime settings are used to redirect requests that target the previous version of Microsoft.ReportingServices.Interfaces to the new version.

8\Common7\IDE\PrivateAssemblies directory. There are three primary elements that Report Designer uses:

- The Data element holds information about data processing extensions.
- The Render element houses information on the rendering extensions.
- The Designer element contains information about the query builders used in Report Designer.

The only time you would modify this file is when you are changing the settings for an existing extension or adding a new extension, such as the Microsoft .NET Data Provider 1.0 for SAP NetWeaver Business Intelligence discussed in Chapter 15. The settings for this file are listed in Table 16.5.

TABLE 16.5: *RSReportDesigner* Configuration File Settings

SETTING	DESCRIPTION
`SecureConnectionLevel`	Specifies the degree of security of the web service connection. Valid values are: 0 = Report Server does not check for a secure connection. 1 = All web service calls that might pass sensitive data and that come over an insecure connection are rejected. 2 = All rendered reports and web service calls require a secure connection. 3 = All calls to the Reporting Services SOAP API must be secure.
`InstanceName`	Do not modify this value; it is an identifier for the preview server.
`SessionCookies`	Specifies whether the Report Server uses browser cookies to maintain session information. Valid values are: True = Use session cookies (default). False = Session data is stored in the reportservertempdb database.
`SessionTimeoutMinutes`	Defines how long a session cookie is valid. The default is 3 minutes.
`PolicyLevel`	Specifies the security policy configuration file. The valid value is `Rspreviewpolicy.config`. You should not modify this value.
`CacheDataForPreview`	When set to True, the Report Designer stores data in a cache file on the local computer. Valid values are True (default) and False.
`Render`	Enumerates the rendering extensions that are available to Report Designer for preview purposes. The set of rendering extensions that are used for preview should be identical to those installed with the Report Server. This setting has three main attributes: `Name` specifies the rendering extension. If you are invoking a rendering extension through code, use this value to call a specific extension. `Type` specifies the fully qualified class name of the extension class and the library name; the names are comma separated.

(CONTINUED)

TABLE 16.5: *RSReportDesigner* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
	`Visible` specifies whether the name appears in any user interface. This value can be True (default) or False. If True, it appears in user interfaces.
`Data`	This lists the data processing extensions that are available to Report Designer. The set of data processing extensions used in Report Designer may be identical to those installed with the Report Server. This has two main attributes: `Name` specifies the data processing extension. `Type` specifies the fully qualified class name of the extension class and the library name; the names are comma separated.
`Designer`	This lists the available query builders that provide a user interface for constructing queries that retrieve data used in reports. By default, there is one visual data tool user interface for all data processing extensions that are included with Reporting Services, but third parties may include custom interfaces.

RSWebApplication

This configuration file stores settings for Report Manager and is saved in the `\Reporting Services\ReportManager` directory. You can modify this file to limit the number of open connections per user, change the default delivery extension, and change the default rendering format for each delivery extension. Table 16.6 lists the settings contained in this file.

TABLE 16.6: *RSWebApplication* Configuration File Settings

SETTING	DESCRIPTION
`ReportServerVirtualDirectory`	Specifies the SOAP endpoint of the Report Server that is used by Report Manager, which is the Report Server virtual directory name. For example, if the Report Server URL is `http://CorpReports/reportserver`, the value for this setting is `reportserver`. You should only use this setting if Report Manager and Report Server are configured to use the same website. If they use different websites, delete the value for this setting and use `ReportServerURL` instead.
`ReportServerUrl`	If the Report Manager and Report Server virtual directories are hosted on separate servers or websites, you can use this setting to specify the SOAP endpoint of the Report Server that is used by Report Manager.

TABLE 16.6: *RSWebApplication* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
	If Report Manager is installed separately on an Internet-facing web server, this value must be set to the fully qualified domain name of a Report Server instance behind the firewall. This value can never be set to `localhost`. `ReportServerURL` and `ReportServerVirtualDirectory` are mutually exclusive. If you specify `ReportServerURL`, you must delete the entry for `ReportServerVirtualDirectory`.
`ReportBuilderTrustLevel`	Specify whether Report Builder is launched in full trust or partial trust. Valid values are `FullTrust` and `PartialTrust`. The default is `FullTrust`. For more information about these trust levels, see Chapter 8.
`PassThroughCookies`	Configures Report Manager to transmit custom authentication cookies. For more information and to view and an example of how to specify the configuration settings, see Chapter 7. This setting is not included in the configuration file by default. To use it, you must add it manually.
`DeliveryUI`	Specifies the delivery extensions that can be used with subscriptions. The extension names that you specify in this section appear as delivery options in the subscription definition pages in Report Manager. Reporting Services includes two default delivery extensions for subscriptions: Report Server Email and Report Server FileShare.
`DefaultDeliveryExtension`	Specifies the default delivery extension. Report Server Email is the default. Valid values include True or False. If more than one extension contains a value of True, the first extension is considered the default extension.
`Configuration`	Specifies configuration options for a delivery extension. You can set a default rendering format for each delivery extension. You can find the valid values for this setting in the render section of the `rsreportserver.config` file.
`MaxActiveReqForOneUser`	Specifies the maximum number of simultaneous and in-progress connections that a single user can open to Report Manager. Once the limit is reached, further connection requests from the user are denied. Valid values are 0 to a maximum integer. 0 indicates no limit on the number of connections. The default is 20. This configuration setting is designed to defend against a denial-of-service attack. This helps because most requests process very quickly, so it is unlikely that a single user will have more than 20 open connections at any given time.

(CONTINUED)

TABLE 16.6: *RSWebApplication* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
	If users are opening more than 15 process-intensive reports at the same time, you might need to increase this value.
`DisplayErrorLink`	Displays a link to the Microsoft Help and Support site that appears in error messages. Users can link to the site to view updated error message content. Valid values include true or false. The default is true.

RSReportServer

The largest of the configuration files, `RSReportServer.config`, contains configuration settings for the Report Server Windows service and the web service and is saved in the `\Reporting Services\ReportServer` directory.

This file contains encrypted connection information and user account values that you can only modify using the Reporting Services Configuration tool or the `rsconfig.exe` command-line utility. The settings contained in this file are listed in Table 16.7.

TABLE 16.7: *RSReportServer* Configuration File Settings

SETTING	DESCRIPTION
`Dsn`, `LogonUser`, `LogonDomain`, `LogonCred`	Specifies encrypted values used internally by a Report Server to connect to a Report Server database. Values for `LogonUser`, `LogonDomain`, and `LogonCred` are created when the Report Server connection is configured to use a domain account. Do not modify these values directly; use the Reporting Services Configuration tool instead.
`ConnectionType`	Specifies the type of credentials that the Report Server uses to connect to the Report Server database. Valid values are `Default` and `Impersonate`. `Default` is used when the Report Server is configured to use a SQL Server login or the service account to connect to the Report Server database. `Impersonate` is used when the Report Server uses a Windows account to connect to the Report Server database.
`InstanceID`	Do not modify this setting. This is an identifier for the Report Server instance, which is based on the SQL Server instance.
`InstallationID`	Do not modify this setting. This is an identifier for the Report Server installation that Setup creates.

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
SecureConnectionLevel	Specifies the degree of security of the web service connection. Valid values are: 0 = Report Server does not check for a secure connection. 1 = All web service calls that might pass sensitive data and that come over an insecure connection are rejected. 2 = All rendered reports and web service calls require a secure connection. 3 = All calls to the Reporting Services SOAP API must be secure.
ProcessRecycleOptions	Specifies whether the ASP.NET worker process is recycled when severe errors occur. Valid values are 0, which allows recycling, and 1, which does not. The default is 0.
CleanupCycleMinutes	Specifies the number of minutes after which old sessions and expired snapshots are removed from the Report Server databases. Valid values range from 0 to the maximum integer. A value of 0 disables the database clean up process. The default is 10.
SQLCommandTimeoutSeconds	This is a reserved setting that is not used.
MaxActiveReqForOneUser	Specifies the maximum number of simultaneous and in-progress connections that a single user can open to Report Manager. Once the limit is reached, further connection requests from the user are denied. Valid values are 0 to the maximum integer. 0 indicates no limit on the number of connections. The default is 20. This configuration setting is designed to defend against a denial-of-service attack. This helps because most requests process very quickly, so it is unlikely that a single user will have more than 20 open connections at any given time. If users are opening more than 15 process-intensive reports at the same time, you might need to increase this value.
DatabaseQueryTimeout	Specifies how many seconds to wait before a connection to the Report Server database times out. Valid values range from 0 to maximum integer. A setting of 0 allows unlimited wait times, so it is not recommended. The default value is 120.

(CONTINUED)

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
RunningRequestsScavengerCycle	Specifies how often, in seconds, that orphaned and expired requests are canceled. Valid values range from 0 to maximum integer with a default value of 60.
RunningRequestsDbCycle	Specifies how often, in seconds, that the Report Server evaluates running jobs to check whether they have exceeded report execution timeouts. It also defines when running job information in the Manage Jobs page of Report Manager is updated. Valid values range from 0 to the maximum integer, with a default value of 60.
RunningRequestsAge	The number of seconds after which a job's status changes from new to running. Valid values range from 0 to the maximum integer with a default value of 30.
MaxScheduleWait	Specifies the number of seconds the Report Server Windows service waits for a schedule to be updated by SQL Server Agent service when Next Run Time is requested. Valid values range from 0 to the maximum integer with a default value of 5.
DisplayErrorLink	Displays a link to the Microsoft Help and Support site that appears in error messages. Users can link to the site to view updated error message content. Valid values include true or false. The default is true.
WebServiceuseFileShareStorage	Specifies whether to store cached reports and temporary snapshots used by the web service on the filesystem. When this is True, temporary data is stored on the filesystem. When this is set to False, temporary data is stored in the reportservertempdb database. The default value is False.
WatsonFlags	Specifies how much information is logged for error conditions that are reported to Microsoft.
WatsonDumpOnExceptions	Do not modify this setting.
WatsonDumpExcludeIfContainsExceptions	Do not modify this setting.
IsSchedulingService	Specifies whether a thread is dedicated to ensuring that schedules in the Report Server database match those in the SQL Server Agent database. Valid values include True (default) and False. Do not modify this value directly. It is changed automatically when you enable or disable Reporting Services features in the SQL Server Surface Area Configuration tool.

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
IsNotificationService	Specifies whether the Report Server dedicates a thread to polling the notification table in the Report Server database to determine whether there are pending notifications. Valid values include True (default) and False. Do not modify this value directly. It is changed automatically when you enable or disable Reporting Services features in the SQL Server Surface Area Configuration tool.
IsEventService	Specifies whether service processes events in the event queue. Valid values include True (default) and False. Do not modify this value directly. It is changed automatically when you enable or disable Reporting Services features in the SQL Server Surface Area Configuration tool.
PollingInterval	Specifies the interval, in seconds, between polls of the event table by the Report Server. Valid values range from 0 to the maximum integer, with a default value of 10.
WindowsServiceUseFileShareStorage	Specifies whether to store cached reports and temporary snapshots used by the Windows service on the filesystem. When this is True, temporary data is stored on the file system. When this is set to False, temporary data is stored in the reportservertempdb database. The default value is False.
MemoryLimit	This defines the maximum memory threshold for the Report Server Windows service. After the threshold is reached, no new processing requests are accepted but requests that are currently in progress are allowed to continue. This value is expressed as a percentage of available memory. Valid values range from 0 to 100, with a default value of 60.
RecycleTime	Specifies, in minutes, how often the application domain will recycle. Valid values range from 0 to the maximum integer, with a default value of 720.
MaximumMemoryLimit	This specifies that the application domain for the Report Server Windows service be recycled when Report Server memory consumption reaches a certain percentage of available memory.

(CONTINUED)

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
	This value is the percentage of available memory that can be used while recycling. The default value is 80. This value should be greater than the value set for `MemoryLimit`.
`MaxAppDomainUnloadTime`	Specifies a wait time, in minutes, during which the application domain is allowed to complete any current activities during a recycle operation. If recycling does not complete during this time period, all processing is stopped and the application domain is immediately restarted. Valid values range from 0 to the maximum integer, with a default value of 30.
`MaxQueueThreads`	Specifies the number of threads used by the Report Server Windows service to process subscriptions and notifications simultaneously on the Report Server. Valid values range from 0 to the maximum integer. The default is 0, which means that the Report Server determines the maximum number of threads. You can manually set the upper limit by specifying an integer as the value.
`UrlRoot`	Used by delivery extensions to compose the URL used to access items stored on the Report Server. You can manually edit this value, or you can use the Reporting Services Configuration tool. The value must be a valid URL address.
`UnattendedExecutionAccount`	Specifies a username, password, and domain used by the Report Server to run a report in unattended mode. These values are encrypted, so you must use the Reporting Services Configuration tool or the `rsconfig.exe` utility to set these values.
`PolicyLevel`	Do not modify this value. It specifies the security policy configuration file. The valid value is `Rssrvrpolicy.config`.
`WebServiceAccount`	Specifies the account used to run the Report Server web service. It is used to confirm the identity of the Report Server web service when performing data-encryption operations. This should be set using the Reporting Services Configuration tool.
`IsWebServiceEnabled`	Do not modify this value directly. This specifies whether the Report Server web service responds to SOAP and URL access requests. This value is set when you enable or disable the web service in the SQL Server Surface Area Configuration tool.

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
FileShareStorageLocation	Specifies a folder on the filesystem for storing temporary snapshots. The default value is `c:\program files\Microsoft SQL Server\MSSQL.x\Reporting Services\RSTempFiles`. It is possible to set this value using a UNC path, but that is not recommended.
Delivery	Specifies default and custom delivery extensions used to distribute reports through subscriptions.
Extension Name	Specifies the delivery extensions that are available to the Report Server. Reporting Services includes default delivery extensions for Report Server: FileShare, Report Server Email, NULL delivery provider, and SharePoint Delivery.
MaxRetries	Specifies the number of times a Report Server will retry a delivery if the delivery fails. The default value is 3.
SecondsBeforeRetry	Specifies the interval of time, in seconds, between each retry attempt. The default value is 900 (which is 15 minutes).
FileShareConfiguration	Specifies settings for subscription-based file-share delivery.
RSEmailConfiguration	Specifies settings for subscription-based e-mail delivery.
SMTPServer	Specifies a string value indicating the address of a remote SMTP server or forwarder and is required for remote SMTP service. This can be an IP address, a UNC name of a server, or a fully qualified domain name.
SMTPServerPort	Specifies the port number that SMTP uses to send e-mail. This is an integer with a default value of 25 (the default SMTP port).
SMTPAccountName	Contains a string value that assigns the Microsoft Outlook Express account name. You can set this value if your SMTP server is configured to use it in some way; otherwise, you can leave it blank.
SMTPConnectionTimeout	Specifies the interval, in seconds, to wait for a valid socket connection with the SMTP service before timing out. The default is 30 seconds. This value is ignored if `SendUsing` is set to 2.
SMTPServerPickupDirectory	Specifies a string value indicating the pickup directory for the local SMTP service. This value must be a fully qualified local folder path (for example, `c:\MailPickup`).

(CONTINUED)

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
SMTPUseSSL	Specifies whether SSL is used to send SMTP messages. When set to 0 (default), then SSL is not used. This setting can be used when the SendUsing element is set to 2.
SendUsing	Specifies which method to use for sending messages. Valid values are: 1 = Sends a message using the local SMTP service pickup directory. 2 = Sends the message using the network SMTP service.
SMTPAuthenticate	Specifies an integer value that indicates the kind of authentication to use when sending messages to an SMTP service over a TCP/IP connection. Valid values are: 0 = No authentication. 1 = Is not valid. 2 = NTLM authentication. The security context of the Report Server Windows service is used to connect to the network SMTP server.
From	Specifies an e-mail address from which reports are sent in the format *name@host.ext*. The address appears on the From line of an outgoing e-mail message. This value is required if you are using a remote SMTP server, and it should be a valid e-mail account that has permission to send mail.
EmbeddedRenderFormats	Specifies the rendering format used to encapsulate a report within the body of an e-mail message. Images in the report are subsequently embedded in the report. Valid values are MHTML and HTML4.0.
PrivilegedUserRenderFormats	Specifies the rendering formats available to users that are assigned to roles with the Manage All Subscriptions task. If this value is not set, all render formats that are not purposely excluded are available for use.
ExcludedRenderFormats	Purposely excludes formats that do not work well with a given delivery extension. By default, HTMLOWC, RGDI, and NULL delivery extensions are excluded for e-mail delivery and file-share delivery. If you try to exclude multiple instances of the same rendering extension, you will get an error when the Report Server reads the configuration file.

TABLE 16.7: *RSReportServer* Configuration File Settings *(CONTINUED)*

SETTING	DESCRIPTION
SendEmailToUserAlias	This value works with DefaultHostName. When this is set to True, the To field is hidden when users configure e-mail delivery for a subscription and Reporting Services defines the user creating the subscription as the recipient of the e-mail. If this is set to False, then the To field is visible and users can specify the e-mail address for the recipient.
DefaultHostName	Specifies a string value indicating the hostname to append to the user alias when SendEmailToUserAlias is set to True. This value can be a DNS name or an IP address.
PermittedHosts	Limits report distribution by explicitly specifying which hosts can receive e-mail delivery. This is a list of HostName elements that have a value of either an IP address or a DNS name. By default, this is empty, which means that there are no restrictions. If you specify DefaultHostName, be sure to include that host as a HostName element of PermittedHosts.
Null	Specifies the NULL delivery provider, used to preload the cache.
Render	Specifies default and custom rendering extensions used in report presentation. Default rendering extensions include NULL, XML, CSV, IMAGE, PDF, RDGI, HTML4.0, HTML3.2, MHTML, and EXCEL.
Data	Specifies default and custom data processing extensions used to process queries. Default data processing extensions include SQL, OLEDB, OLEDB-MD, ORACLE, ODBC, and XML.
Semantic Query	Specifies semantic query processing extension used to process report models. The semantic query processing extensions included with Reporting Services provide support for SQL Server relational data and Analysis Services multidimensional data. Do not modify this value.
Security	Specifies default and custom authentication extensions used to support authentication.
EventProcessing	Specifies default event handlers. Do not modify this value.

Tips and Techniques

This section gives you a few tips and techniques for monitoring and configuring Reporting Services.

Setting a Measurement Baseline

You will never know if your system is running slower than normal unless you know what *normal* is. A measurement baseline defines *normal* for you: it shows you the resources (memory, CPU, etc.) that Reporting Services consumes under typical circumstances. You create the measurement baseline before putting your system into production so that you have something to compare your readings to later on.

To create an accurate measurement baseline, the first thing you need is a test network with just your Report Server and one or two client machines. You limit the number of machines involved because all networks have broadcast traffic, which is processed by all the machines on the network. This broadcast traffic can throw your counts off — sometimes a little, sometimes quite a bit. You may instead want to consider shutting down as many machines as possible and generating your baseline off-hours if your budget does not allow for a test network.

You can then start your baseline. The Windows counters mentioned at the beginning of this chapter as well as the preset Reporting Services counters should provide an accurate baseline with which you can compare future readings. Then you can move to the next technique.

Data Archiving and Trend Tracking

You need to save, or archive, your records. One of the primary reasons to do so is to back up requests for additional equipment. For example, if you ask for funds to buy more memory for the Report Server but don't bring any proof that the system needs the RAM, you may not get the money to make the purchase. If you bring a few months' worth of reports, however, and say, ''After tracking the Report Server usage for a time, we've found this. . .'' management may be far more willing to give you the funds you need. Using archived data in such fashion is known as *trend tracking*.

One of the most valuable functions of using your archived data for trend tracking is proactive troubleshooting — that is, anticipating and avoiding problems before they arise. Suppose you added 50 new users to your network about three months ago and are about to do it again. If you archived your data from that period, you would be able to recall what those 50 users did to the performance of the Report Server, and you could compensate for it. On the other hand, if you threw that data away, you might be in for a nasty surprise when your system unexpectedly slows to a crawl.

Manually Configuring Available Memory

The Report Server is implemented as a Windows service and a web service. Each service handles different operations. The Windows service processes scheduled operations and handles encryption. The web service processes on-demand report requests. The method for configuring available memory is different for each service.

For the Web Service

`ASP.NET` controls how memory is used by the web service, so the memory settings are configured in the `machine.config` file. The `memoryLimit` attribute of the `processModel` element specifies the maximum percentage of memory that a worker process can consume before IIS launches a new

worker process and reassigns any existing requests. This setting is critical because it affects the way that unused information in memory is cleared out through a process called *garbage collection*.

The default value is 60 percent of available memory, which is a good start, but it may not be what you need. To verify, you can monitor the Pages/Sec counter of the Memory object and the Private Bytes (aspnet_wp) counter of the Process object in Performance Console.

If the Pages/Sec counter is greater than 5 on a sustained basis and the Private Bytes (aspnet_wp) counter is nearing the limit, you may need to increase the value.

If you have 4GB or more of RAM in your system, you will either need to lower this value or use the `/3GB` switch in your `boot.ini` file. This is because 60 percent of 4GB is 2.4GB of RAM, and the `ASP.NET` process will only use 2GB of RAM by default. If `ASP.NET` uses more than 2GB of RAM in this circumstance, your users will see an `OutOfMemoryException` error. Decreasing the `memoryLimit` to a value that will not use 2GB of memory can help, or you can use the `/3GB` switch, which allows applications to access more memory. With the `/3GB` switch, `ASP.NET` can use safely use 1,800 MB of RAM.

For the Windows Service

The settings for configuring the way the Windows service uses memory are housed in the `RSReportServer.config` file. These setting are:

`MemoryLimit` This setting specifies the maximum threshold after which no new processing requests are accepted. This is a percentage of available memory with a default value of 60.

`MaximumMemoryLimit` This specifies a maximum percentage of memory that the application domain is allowed to consume. If this percentage is reached, the application domain is recycled. The default value is 80; it should be larger than the `MemoryLimit` setting.

Reporting Services will do a good job of monitoring the amount of physical memory for you, but if you want to use virtual memory as well, you can set the values to 100.

Real World Scenario

Case Study: Chocolaty Goodness Candies

We had a fairly nice Report Server at CGC; among other things, it had 8GB of physical RAM. We had to give it some power because we started serving up some complex reports for several hundred users.

The server worked great for a while after installation, but then the calls started coming in. Every day or so we would get a complaint that a user had received an error that the server was out of memory.

I didn't quite realize what was happening at first because it seemed preposterous that a server with 8GB of RAM could run out of memory. During my research, though, I discovered that it was not the server, but the `ASP.NET` process that was running out of memory.

Because I had enough RAM in the server, I configured the system to use the `/3GB` setting by modifying the `boot.ini` and the memory errors never came up again.

The Bottom Line

Use performance counters. Using Performance Console, you can monitor objects and counters. An object is a part of the system, such as the processor or the physical memory. A counter displays the statistical information about how much that object is being used.

Reporting Services installs performance counters for both the Windows and web services.

> **Master It** Some of your users have been complaining that the Report Server is not returning reports in a timely manner, and some of the users are getting error messages. You want to find out how many requests for reports are being processed by the web service, how many of those requests are successful, and how many fail. What counters should you monitor?

Specify settings in each of the configuration files. All of the settings that Reporting Services requires to run are stored in configuration files saved on the hard disk. The configuration files all contain XML data, so you can use any XML editor or text editor to modify the files, though some of the settings are encrypted, which means you must use Reporting Services tools to modify them.

> **Master It** What are the primary configuration files, where are they stored, and what is each one used for?

Monitor and configure Reporting Services. Before you can monitor Reporting Services to see if it is running slower than normal, you need to know what normal is. Set a measurement baseline to define normal. You should then continuously monitor Reporting Services to make sure it is running as expected and keep the logs for trend tracking and archiving. If the system is running slow, you may have to adjust the amount of available memory.

> **Master It** Some of your users are complaining that they are receiving an out-of-memory exception when they try to use the Report Server. You have 4GB of RAM, so you know that the system is not actually running out of memory. The users who get this error are all using service provided by the web service. What is the problem and what can you do to fix it?

Appendix A

Chapter 1

Define a reporting infrastructure Simply put, a reporting infrastructure combines one or more data sources with a mechanism for displaying the data in a format that makes sense to end users.

Master It What are the two primary components of a reporting infrastructure?

Solution The two main components of any reporting infrastructure are the data source and the mechanism that displays the report in a human-readable format.

Figure out why an enterprise-reporting infrastructure is necessary Just because you have data does not mean that you need an enterprise-reporting infrastructure. You may only have a small amount of data that you report on infrequently, and the reports may be fairly simple. This scenario may not require an enterprise-reporting infrastructure. If you have a lot of data from a number of data sources that you require complex reports from, then you should seriously consider an enterprise-reporting infrastructure.

Master It You work for a mid-sized company that has a Human Resources database on an Oracle 9*i* system and an Accounting database on SQL Server 2005. You need to generate salary and bonus reports for upper management based on both of these databases. Does this require an enterprise-reporting infrastructure?

Solution The answer is, most assuredly, yes. You may be able to conjure up some reports in something like Access or Visual FoxPro, but these would be difficult to develop, maintain, and generate on a regular basis. Using an enterprise-reporting infrastructure, you can perform all of these functions much more efficiently.

Decide if you need to use Reporting Services as your reporting infrastructure Reporting Services can easily handle any reporting task you can throw at it, so functionality is not the question here. The real question is how much will Reporting Services cost to implement. Talk with your vendor for actual pricing, but you will find that, with the right licensing, Reporting Services is a definite contender.

Master It You work for a mid-sized company that has an inventory management database on a SQL Server 2005 server. You need to create a report for your customers that shows your current inventory. Your customers need to access this report over the Internet. Can you use Reporting Services to meet your needs?

Solution Yes, you can use Reporting Services to meet your needs in this scenario. All you need to do is make your Report Server available over the Internet and set the proper

security (which we will discuss later in the book), and your customers will be able to view your inventory reports.

Chapter 2

Analyze the current business model.

Master It When you are analyzing the current business model, what are some of the types of business processes you should look for?

Solution Generally speaking, there are four types of business processes you should look for:

- Information flow
- Communication flow
- Service and product lifecycles
- Decision making

Determine new business requirements.

Master It A scope document is one of the most important documents you will come up with during your project. What is it used for?

Solution A scope document accomplishes four primary tasks:

- It is used to make sure that all of the decision makers and stakeholders involved in the project have a clear understanding of the goals for the project.
- It lays out the risks and alternatives involved in the project.
- It identifies the resource requirements for the project.
- It is used to present a budget for approval.

Plan for future business requirements.

Master It What are some of the important factors to consider when planning for future business requirements?

Solution First, remember that any projection you make about the future is just a guess, so try to make the most educated guess possible. Also, take the company's goals for projected growth into account. It is also important to remember not to act on the future projections until they come to pass; otherwise, you may end up wasting money.

Chapter 3

Plan and develop role-based security as part of your underlying Reporting Services infrastructure.

Case Study: AlsoRann Industries

You have been assigned to develop a new Reporting Services architecture for AlsoRann. To start the project, you schedule meetings and interviews with the management and executive staff to get a better understanding of the security requirements for the new infrastructure.

In the course of your analysis, you find that AlsoRann is a publicly held company, so they need to conform to the Sarbanes–Oxley requirements. They also have a public website on which customers are allowed to store personal information, such as addresses and credit card information, so the company needs to adhere to the requirements of the various credit card companies and regulatory bodies as well.

Master It In a similar situation to this case study, consider how you would go about validating your security plan. Who might you consult with to ensure that your plan is valid and meets all the necessary criteria?

Solution To ensure compliance with all these regulations, you will need to validate every security decision against all the various requirements. You should consider meeting with upper management and your company's legal counsel to verify your plan. You may even have the external regulatory bodies, such as the FDA, validate your decisions as well.

Understand the role-based security model used in SQL Server Reporting Services.

Master It Describe tasks, permissions, role definitions, and role assignments.

Solution Permissions provide access to specific report server functions. A task is a set of predefined permissions. Role definitions are a named collection of tasks. Role assignments are a combination of role definitions, securable items, and user or group accounts.

Create and modify role definitions A role definition is a named collection of tasks that specify which tasks a user is allowed to perform in connection with a folder, report, or other item. While Reporting Services includes several predefined roles to accommodate various categories of users, you may have to create additional roles to meet your security needs.

Master It Using Management Studio, create a new item-level role definition called Report Reviewer.

Solution

1. Open Management Studio and expand the Report Server in Object Explorer.
2. Expand the Security folder.
3. Right-click the Roles folder and select New Role to open the New User Role properties sheet.
4. In the Name text box, type **Report Reviewer**. Add a description if you wish.
5. In the task pane, mark the check box next to the tasks you wish members of this role to perform.
6. Click OK. The role definition appears in the Roles folder.

Create and modify role assignments Reporting Services provides default security through role assignments that are configured during setup. These role assignments define access for local administrators. Typically, you will create additional role assignments for folders or reports that relate to specific users or groups; for example, you may want to add role assignments to a group's folders that give individuals in that group the capability to manage those folders and their contents.

Master It Using Report Manager, create a new role assignment.

Solution

1. Open Report Manager and navigate to the Home folder.
2. Select the Properties tab to open the Security page for the Home folder.
3. Click New Role Assignment. Note that you can open the New Role page by clicking the New Role button. This is useful if you don't want to use any of the existing role definitions.
4. In the Group or User Name text box, type the name of a valid user or group in the ***DomainName**UserName*** or format.
5. Check the check box for the Publisher role. You can also select multiple roles to apply.

Chapter 4

Communicate effectively with your users If you do not communicate effectively with your users, your project is doomed to fail. You need to get input and feedback from the user community to make sure they know how to use the system and ensure that they are satisfied with it.

Master It How often should you communicate with your users and in what settings?

Solution Communicate with your users as often as possible. You should have a project status meeting at least once a week, and possibly more often depending on your company's needs. You can also communicate with users one on one, over the telephone, and via e-mail.

Provide education for your users Users will not be happy with the system if they do not know how to use it. To make sure they understand the system, you need to provide education.

Master It What are some of the key differences between standard user documentation and custom courseware used in teaching a class?

Solution Custom courseware differs from standard documentation in several key ways:

- Courseware is a series of timed modules. Documentation does not need to be timed.
- Courseware generally contains assessment questions with answers; documentation does not.
- You need two editions of courseware: a student edition and an instructor's edition of the courseware with all of the answers and instructor notes. You only need a single edition of documentation.

Develop and use service-level agreements A service-level agreement (SLA) is a contract between the IT department and the user community that defines the system availability. This is a protection for you and the users.

Master It What components do you need to define in order to craft a service-level agreement?

Solution You will need to include a number of key components in your SLA in order to make it effective. You will need to define:

- The parties to the SLA
- The services provided
- Service volume
- Timeliness
- Availability
- Reliability
- Compensation
- Measurement

Chapter 5

Understand the theory of change To effectively manage change for your organization, you must understand some of the theories behind change.

Master It What are some theories of change and what do they consist of?

Solution We discussed three theories of change:

- The unfreeze/change/refreeze theory
  - Unfreeze
  - Change
  - Refreeze
- ADKAR
  - Awareness
  - Desire
  - Knowledge
  - Ability
  - Reinforcement

Understand the skills needed to effect change Knowing the theories of change is not enough to make change happen. You need to master some skills in order to effect change.

Master It What are some of the skills that you need to acquire to manage change in your organization?

Solution You should acquire these important skills to manage change:

- Political
- Analytical
- Business
- People
- System

Choose a change-management strategy Once you know what change is all about and you have the skills to manage it, you need to choose a strategy for managing change throughout your project.

Master It What are three main strategies for managing change in an organization?

Solution The three main strategies we discussed are:

- Empirical-rational
- Normative-reeducative
- Power-coercive

Manage ongoing change After the project is complete and the changes have all been made, you must continue to manage ongoing change. If you do not manage changes to the system, your entire reporting system could crash — which would cause you to lose user confidence.

Master It What should you include in a change request form?

Solution Some of the basic pieces of information you will need are:

- The date of the proposed change
- The system(s) affected by the change
- A detailed description of the proposed change
- A detailed rollout plan
- A detailed rollback plan in case the change causes problems
- A list of users to notify about the change

Chapter 6

Gain a better understanding of OLAP Online analytical processing (OLAP) allows you to create and manage multidimensional structures called cubes that contain data that has been collected from other sources.

Master It What are the two types of tables that make up a cube, and what do they contain?

Solution The two types of tables are dimension and fact tables. Dimension tables store dimensions, which are used to organize data by topics of interest. Fact tables contain measures, which are pieces of data that you want summarized by dimension.

Gain a better understanding of data mining OLAP is just one of the tools available for data analysis. SQL Server Analysis Services also provides data-mining tools to enhance the data analysis functions.

Master It Describe data mining and give an example of when it would be useful.

Solution Data mining is the process of looking for patterns in data. There are many circumstances when this would be useful; the one we used earlier in this chapter is a good example.

Management needs to make business decisions based on the data in those cubes. To make business decisions, management looks for patterns in the data. For example, management may need to know how many men's leather coats were sold in the northeastern

United States last November so they can make preparations for the coming November. They may also look at production quotas, sales by store, materials usage in manufacturing, and similar factors.

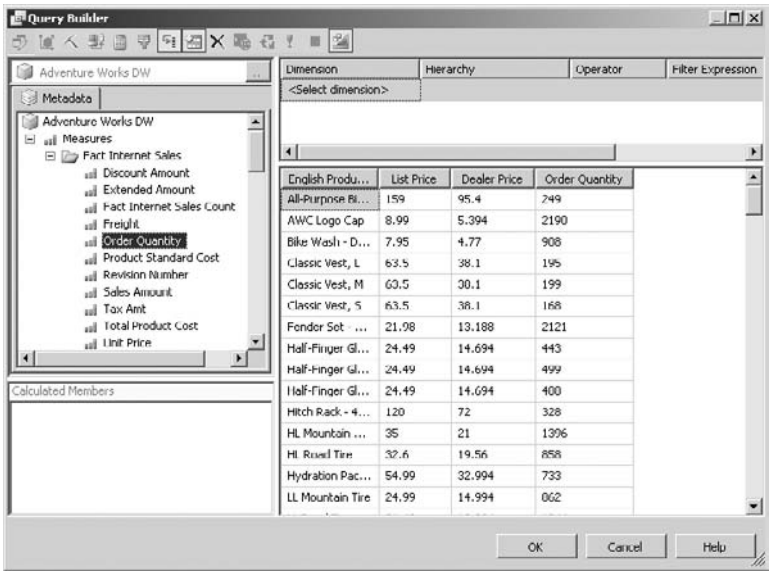
The data-mining tools included with Analysis Services are necessary for this process because cubes do not show patterns by themselves. You need to create and train a data model to find these patterns.

Learn how to base a report on a cube or data-mining model Once you have created a cube or a mining model, you need to be able to create a report based on these data sources so that you can get the information contained in them to the management staff.

Master It Create a report based on the Reporting Services Example cube that shows the list price, dealer price, and order quantity grouped by product name.

Solution

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio to open BIDS.
2. Select File ➢ New ➢ Project.
3. Select Business Intelligence Projects from the Project Types list.
4. Select Report Server Project Wizard from the Templates list.
5. Type **Second Cube Report Example** in the Name text box (note that the Solution Name box is filled in for you).
6. Click OK to start the wizard.
7. On the Welcome screen, click Next.
8. On the Select The Data Source screen, type **Cube DS** in the Name box.
9. Select Microsoft SQL Server Analysis Services from the Type drop-down list box.
10. Click the Edit button to create a connection string.
11. Type **localhost** in the Name box (or the name of the server Analysis Services is installed on if it is not your local server).
12. Select Reporting Services Example from the drop-down list box under Select or enter a database name.
13. Click OK to return to the wizard.
14. Click Next to continue.
15. Click the Query Builder button at the top of the Design The Query screen.
16. In the Metadata pane, expand DimProduct.
17. Drag the English Product Name, List Price, and Dealer Price dimensions to the area labeled ''Drag levels or measures here to add to the query.''
18. In the Metadata pane, expand Measures | Fact Internet Sales.
19. Drag the Order Quantity measure to the query area.



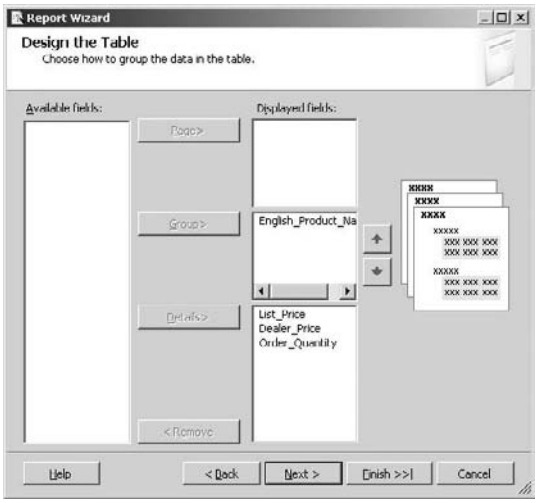
20. Click OK to return to the wizard. Notice that the Query Builder has constructed an MDX language query for you.

21. Click Next to continue.

22. Select the Tabular option on the Select Report Type screen and click Next.

23. On the Design The Table screen, select the English Product Name field from the Available Fields list and click the Group button.

24. Add the remaining fields to the Details area by selecting them one at a time and clicking the Details button.



25. Click Next to continue.
26. Accept the defaults on the Choose The Table Layout screen and click Next.
27. Select a style on the Choose The Table Style screen and click Next.
28. Accept the defaults on the Choose The Deployment Location screen and click Next.
29. Type **Second Cube Report Example** in the Report Name box and click Finish to create the report.

Chapter 7

Know what types of deployments are available and how to use them Two types of deployment available: standard and scaled-out. Standard deployments involve a single Report Server and a single database server. Scaled-out deployments involve multiple Report Servers using a single database server.

Master It What are the main considerations for implementing a scaled-out deployment?

Solution You need to ensure three things before implementing a scaled-out deployment:

- All of the Report Servers must be the same version and service pack level.
- All of the servers involved must be in the same domain or a trusted domain.
- If you are running Service Pack 2, all of the servers must be in the same mode of operation, either native or SharePoint integrated.

Know how to integrate with SharePoint SharePoint is collaboration software that makes it easy for users to work together on projects by sharing documents and graphics, joining discussion groups, and the like. Reporting Services can be configured to work with SharePoint as a separate application on the same machine, or they can be integrated to work together.

Master It Reporting Services cannot be installed in integrated mode, so what steps do you need to take to configure SharePoint integrated mode?

Solution Here are the steps to configure your server for SharePoint integrated mode:

1. Install SQL Server 2005 Service Pack 2.
2. If the Report Server and SharePoint web application are installed on separate computers, install the SharePoint object model on the Report Server.
3. Start the Reporting Services Configuration tool and connect to the Report Server instance that you want to configure for SharePoint integration.
4. If this is a new installation, select the default website and enter a name for the virtual directory on the Report Server Virtual Directory page. Skip the page for configuring the Report Manager virtual directory because you cannot use Report Manager with a Report Server that is configured for SharePoint integrated mode.
5. On the Database Setup page, create a new Report Server database in SharePoint integrated mode by selecting the Integrate With SharePoint Products And Technologies option when you create the database.
6. Back up the encryption key.

7. If you are working with a new installation of Reporting Services, you should also configure the unattended report processing account if you want to support unattended report execution. Optionally, configure the Report Server for e-mail delivery.
8. Close the Reporting Services Configuration tool.

Configure Reporting Services components after installation Most of the settings that you need to configure to get Reporting Services up and running in production can be made using the Reporting Services Configuration tool. You will need to configure the more advanced settings manually.

Master It What are some of the advanced settings that you need to configure manually?

Solution Some of the advanced settings that you need to configure manually are:

- Custom authorization cookies
- Advanced e-mail settings
- Internet access

Automate deployment tasks When you need to install Reporting Services on a number of servers in your organization, it is usually best to automate the process. Reporting Services supports automation for several steps in the deployment process.

Master It You need to automate the Reporting Services setup program to install Reporting Services, Report Manager, and the Reporting Services command-line tools. You want to use the default configuration, and you need Reporting Services to start automatically when Windows starts. You are installing Reporting Services on the same server as the database engine and you are connecting to the default instance. How would you do this?

Solution There are two steps to accomplish this goal. You need to execute `Setup` from the command line using this format:

```
Setup /settings c:\customsetup.ini /qn
```

Before you run the `Setup` command, create the custom INI file. To install all the options listed at the outset, the INI file would look like this:

```
[Options]
INSTANCENAME=MSSQLSERVER
   ADDLOCAL=RS_Server,RS_Web_Interface,RS_tools
RSAUTOSTART=1
RSCONFIGURATION=Default
RSSQLLOCAL=1
```

Chapter 8

Configure your systems to support ad hoc reporting Before your users can start creating ad hoc reports, you need to configure the systems to run Report Builder.

Master It You have installed Reporting Services on your network and you need to allow the sales staff to create ad hoc reports using Report Builder. All of the sales staff employees

are members of a Windows security group named Sales in the Corp domain. What do you need to do to allow the sales staff access to Report Builder?

Solution To give the Corp\Sales group access to use Report Builder, follow these steps:

1. Open Report Manager and navigate to the Home folder.
2. Select the Properties tab to open the Security page for the Home folder.
3. Click New Role Assignment.
4. In the Group Or User Name text box, type **Corp\Sales**.
5. Check the box for the Report Builder Role.
6. Click OK to save the new role assignment and return to the Securities page.
7. Click Site Settings.
8. In the Security section, select Configure Sitewide Security to open the System Role Assignments page and click New Role Assignment.
9. In the Group Or User Name text box, type **Corp\Sales**.
10. Check the box next to the System User role.
11. Click OK.

Create and deploy report models A report model is a layer of metadata in between the database and Report Builder that is used to describe the underlying data source in user-friendly terms. Models can be based on SQL Server 2000 databases, SQL Server 2005 databases, Oracle 9.2.0.3 and greater databases, and SQL Server 2005 Analysis Services cubes. You can use Model Designer, Report Manager, and Management Studio to create report models; the only caveat is that Model Designer cannot be used to create models based on Analysis Services cubes.

Master It Now that your sales staff has access to Report Builder, they want to be able to create reports based on the data in the Sales database. How would you create a model that presents all of the data in the Sales database for them to use?

Solution First, you need to create a model project in Business Intelligence Development Studio:

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Business Intelligence Development Studio.
2. On the File menu, point to New, and then click Project.
3. Click Report Model Project in the Templates list.
4. In the Name text box, type **SalesModel**.
5. Click OK to create the model project.

Next, you need to create a data source to connect to the Sales database:

1. In Solution Explorer, right-click Data Sources and click Add New Data Source.
2. On the Welcome screen, 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, type your server name in the Server Name text box.
5. Select Use Windows Authentication.
6. Select Sales from the Select Or Enter A Database Name list box.
7. Click OK to create the connection.
8. Verify that the ''Create a Data Source Based on an Existing or New Connection'' option is selected, verify that ***servername*.Sales** is selected in the Data connections list box, and click Next.
9. In the Data source name box, type **SalesModelDS**, and then click Finish.

The next step is to create a data source view:

1. In Solution Explorer, right-click Data Source Views and click Add New Data Source View.
2. On the Welcome screen, click Next.
3. On the Select A Data Source screen, select the SalesModelDS data source you created and click Next.
4. On the Select Tables And Views screen, click the ≫ button to add all available objects to the view.
5. Click Next.
6. Type **SalesModelDV** in the Name text box.
7. Click Finish.

Now that you have a data source and a data source view in your project, you are ready to create a report model:

1. In Solution Explorer, right-click Report Models, and then select Add New Report Model.
2. On the Welcome screen, click Next.
3. On the Select A Data Source View screen, select the `SalesModelDV.dsv` file that you just created and click Next.
4. Accept the defaults on the Select Report Model Generation rules screen and click Next.
5. On the Update Statistics screen, make sure that Update Model Statistics Before Generating is selected, and then click Next.
6. In the Name box, enter **SalesModelRM** and click Run.
7. To exit the wizard, click Finish.

When you have finished creating and configuring the model, you must deploy it to the server:

1. In Solution Explorer, right-click SalesModel and click Properties.
2. Change the `TargetServerUrl` property to match your server name: **`http://servername`/Reports**.
3. Click OK.
4. From the Build menu, click Deploy SalesModel.

Use Report Builder to work with ad hoc reports Report Builder is a `.NET` Framework 2.0 application that users can install using ClickOnce technology and run on their local machine to create and design reports. The reports they design can be saved to the server for future use, or discarded after a single session.

Master It To create a report you need to choose a report layout. What default report layouts are available?

Solution There are three report layouts available to you out of the box:

Table The table layout is the most basic report layout. This layout shows a table in which data from the data source is presented in rows and columns, much like an Excel spreadsheet.

Matrix Like the Table layout, the Matrix layout is arranged in columns and rows. However, with the Matrix layout you can present static or dynamic columns and rows. When the user runs the report, the dynamic columns will expand across the page, presenting as many columns as there are groups in the report. Dynamic rows expand down the page. The data presented to the user are summary values of the data based on the intersection of rows and columns. In this way, a matrix layout works much like a pivot or cross-tab table in Excel. This is useful for creating a report based on an Analysis Services cube.

Chart As the name implies, the chart layout presents a graphical chart to the user viewing the report. The individual creating the report can choose from several different styles, such as area, bar, column, doughnut, line, and pie. Charts also allow a variety of formatting options, like color and 3D effects.

Chapter 9

Know how subscriptions are processed The scheduling and delivery processor makes sure that reports are generated and delivered on schedule. So, to fully understand subscriptions, you need to know what this processor does when it runs a subscription.

Master It What are the steps that the scheduling and delivery processor takes to process a subscription?

Solution These are the steps that the scheduling and delivery processor takes when processing a subscription:

1. When a user creates a subscription, they define a schedule that defines the date and time that the report will be run, a snapshot will be refreshed, or a cache will expire.

2. The Report Server saves the schedule information in the Report Server database.
3. The Report Server creates a job for the SQL Server Agent to run based on the schedule information provided.
4. When the SQL Server Agent runs the job on the date and time specified in the schedule, it creates an event that is added to a queue maintained by Reporting Services.
5. The Report Server Windows service polls the queue for new events every 10 seconds and processes new events as they are entered.

Create standard subscriptions Standard subscriptions create a single instance of a report for all users to view. The values that the subscription uses to create the report cannot be changed at runtime.

Master It You have created a report that displays company sales figures. The sales manager would like to subscribe to the report. You do not need to modify the report parameters at runtime, so you decide to recommend a standard subscription that uses the file-share delivery method. How would you have the sales manager create the subscription?

Solution Users create standard subscriptions for themselves, so you should have the sales manager follow these steps to create the subscription (assuming the folder name is `AdventureWorks Sample Reports`):

1. In Report Manager, on the Home page, click the AdventureWorks Sample Reports link and open the Company Sales report.
2. Click the Subscriptions tab, and then click New Subscription.
3. For the method of delivery, select Report Server File Share from the Delivered By list box.
4. Type **Company Sales Figures** in the File Name text box.
5. Ensure that Add A File Extension When The File Is Created is selected. This causes the report to be saved with a three-character file extension that is determined by the selected rendering format.
6. Select Adobe Acrobat (PDF) from the Render Format drop-down list.
7. In the Path text box, type a Universal Naming Convention (UNC) path to a shared folder on the network (e.g., ***server*****Subscriptions)**.
8. Select the Increment File Names As Newer Versions Are Added option to leave existing report files intact and create new files when the subscription runs.
9. Click the Select Schedule button to schedule the report.
10. Define the appropriate schedule.
11. Click OK to create the schedule and return to the previous page.
12. In the User Name and Password text boxes, enter your own username and password using the format ***domain**username*** for the username.
13. Click OK to create the subscription.

Create data-driven subscriptions Data-driven subscriptions can create multiple reports in multiple formats, each having a unique set of data. They do this by retrieving subscription settings from a data source.

Master It You have created a report that displays company sales figures. Several of the employees in the sales department would like to get this information on a regular basis, but they need it in different formats. You realize that the only way to satisfy this requirement is with a data-driven subscription. How would you create the subscription?

Solution Administrators create data-driven subscriptions because they require a subscriber data source. Because users do not generally have permission to create databases or tables in an existing database, an administrator has to create the subscriber data source. Here is how to create a data-driven subscription:

1. Start Management Studio, and open a connection to a Database Engine.
2. Click the New Query button on the toolbar.
3. Enter this query:

```
USE SubscriberData
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('1','289', 'Valdez', '1', 'IMAGE')
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('2', '284', 'Alberts', '1', 'MHTML')
INSERT INTO [dbo].[UserInfo] (SubscriptionID,
EmployeeID, LastName, FileType, Format) VALUES
('3', '275', 'Blythe', '1', 'PDF')
GO
```

4. Click the Execute button on the toolbar.

Now that you have a database and table to hold the subscriber data, a you can create a data-driven subscription based on the Employee Sales Summary report:

1. On the Home page in Report Manager, click the AdventureWorks Sample Reports link and open the Employee Sales report.
2. Click the Subscriptions tab, and then click New Data-Driven Subscription.
3. Type a description for the subscription if you like; this step is optional.
4. Select Report Server File Share from the delivery method drop-down list.
5. Make sure that Specify For This Subscription Only is selected for the recipient information data source and click Next.
6. In Type, select Microsoft SQL Server as the data source type.
7. In Connection String, type the following connection string: **data source=localhost; initial catalog=SubscriberData**.

8. Click Credentials Stored Securely On The Report Server.
9. Enter your username in the User Name box in the ***domain**username*** format. Then enter your password in the Password text box.
10. Check the ''Use as Windows Credentials When Connecting to the Data Source'' box.
11. Click Next.
12. Enter this query in the query pane:

```
SELECT SubscriptionID, EmployeeID, LastName,
FileType,  Format  FROM  UserInfo
```

13. Click Next.
14. Under File Name, select the ''Get the Value from the Database'' option and select LastName from the list of fields.
15. Under File Extension, select the Get the Value from the Database option and select FileType from the list of fields.
16. Under Path, select the Specify A Static Value option and enter the UNC path to an existing share. It should look like this: ***servername*\Subscriptions**.
17. Under Render Format, select the ''Get the Value from the Database'' option and select Format from the list of fields.
18. Under User Name, select the Specify A Static Value option and enter your username in the text box. It should look like ***domain**username***.
19. Under Password, select the Specify A Static Value option and enter your password in the text box.
20. Under Write mode, select the Specify A Static Value option and select AutoIncrement.
21. Click Next.
22. Use the default parameter values for Month and Year.
23. Under Employee, select the ''Get the Value from the Database'' option and select EmployeeID from the list of fields.
24. Click Next.
25. Select the On A Schedule Created For This Subscription option.
26. Click Next.
27. Define the appropriate schedule.
28. Click Finish.

Know how to manage subscriptions After your users start creating subscriptions, they will come to rely on them to perform their day-to-day duties. You will have to manage those subscriptions to make sure that they are running as expected. The users will be able to check the status of their own subscriptions in Report Manager, but you must fix any issues that may arise.

Master It One of your users is complaining that they are not receiving a report from a subscription on a regular basis. The subscription had been working fine until just a few days earlier. On investigation, you find that the subscription has become inactive. What should you check?

Solution You should check the most common problems that cause subscriptions to become inactive:

- Removing the delivery extension that the subscription uses to deliver reports
- Changing the credential settings of the data source to integrated or prompted values
- Changing a parameter name or data type in the report that is being subscribed to
- Changing the execution mode of a report, e.g., so that it runs on a schedule instead of a snapshot event

Know how to use subscriptions to preload the report cache Reporting Services can cache reports to accelerate access times. Reporting Services processes the report normally the first time it is accessed; subsequent requests are filled from the cache. You can preload the cache so that all user requests are filled from the cache by creating a subscription that targets the Null Delivery Provider as the report destination.

Master It The users that have subscribed to the Company Sales report are complaining that it is too slow. You decide to have the report cached to speed up access, but you do not want the first user who accesses the report to experience lag time. How can you preload the report into the cache?

Solution To preload the report cache, you need to create a subscription that uses the Null Delivery Provider as the report destination. Follow these steps to create a subscription to preload the report cache:

1. In Report Manager, navigate to the Contents page, and then navigate to the report you want to create a subscription for.
2. Click the report, click the Subscriptions tab, and then click New Data-Driven Subscription.
3. Optionally type a description for the subscription.
4. From the Specify How Recipients Are Notified list, select Null Delivery Provider.
5. Specify a data source type and then click Next to configure the data source.
6. Specify the connection type, connection string, and credentials for accessing the data source that contains subscriber data. The following example illustrates a connection string used to connect to a SQL Server database called Subscribers:

```
data source=servername;initial catalog=Subscribers
```

7. Click Next.
8. Specify the query or command that retrieves subscriber data. Optionally increase the timeout period for queries that take a long time to process. For example:

```
Select * from UserInfo
```

9. Click Validate. The query must be validated before you continue. When the Query Validated Successfully message appears, click Next.
10. Because you cannot configure delivery extension settings for the Null Delivery Provider, click Next.
11. Specify report parameter values for the subscription, and click Next.
12. Specify when the subscription is processed. Do not choose When The Report Data Is Updated On The Report Server. That setting is for snapshots only. If you want to use a preexisting schedule, select On A Shared Schedule. Or, to create a custom schedule, click On A Schedule Created For This Subscription and then click Next. Configure the schedule and then click Finish.
13. Configure the Execution options as shown in the steps that follow. On the report page, click the Properties tab.
14. In the left frame, click the Execution tab.
15. On the page, select Render This Report With The Most Recent Data.
16. Choose one of the following two cache options and configure the expiration as follows:
    - To make the cached copy expire after a particular time period, click Cache A Temporary Copy Of The Report. Expire Copy Of Report after A Number Of Minutes. Then type the number of minutes for report expiration.
    - To make the cached copy expire on a schedule, click Cache A Temporary Copy Of The Report. Expire Copy Of Report On The Following Schedule. Then click Configure, or select a shared schedule to set a schedule for report expiration.
17. Click Apply.

Chapter 10

Manage the Windows and web services Reporting Services is made up of a Windows service and a web service. The Windows service performs initialization, encryption, database maintenance, and scheduling and delivery functions. The web service processes ad hoc reports and provides developers with a programming interface so they can integrate Reporting Services into their applications.

Master It You work for a large organization that uses Reporting Services heavily. After taking inventory of your hardware, you are convinced that you need to distribute Reporting Services functionality across multiple servers. How can you do this?

Solution You can install Reporting Services on two different servers and then disable the Windows service function on one of the servers and disable the web service on the second server.

On the first server, follow these steps to disable the Scheduling and Delivery functions of the Windows service:

1. Click Start ➢ All Programs Microsoft SQL Server 2005 ➢ Configuration Tools ➢ SQL Server Surface Area Configuration.
2. Click Surface Area Configuration For Features.

3. Expand Reporting Services and click on Scheduled Events And Report Delivery.
4. Uncheck the box next to Enable Scheduled Events And Report Delivery.
5. Click Apply.

Here are the steps to disable the web service on the second server:

1. Click Start ➢ All Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ SQL Server Surface Area Configuration.
2. Click Surface Area Configuration For Features.
3. Expand Reporting Services and click on Web Service And HTTP Access.
4. Uncheck the box next to Enable Web Service And HTTP Access.
5. Click Apply.

Back up and restore the Report Server database There are a number of valid reasons to back up the databases that support Reporting Services, such as theft, natural disaster, and hardware failure. You should back up two databases: the ReportServer database and the ReportServerTempdb database. The ReportServerTempdb database does not hold persistent data, so you only need to back up the structure. You need to back up the data and the structure of the Report Server database.

Master It You have just installed Reporting Services on a new server and you decide to perform a backup to protect your hard work. You try to perform a transaction log backup of your ReportServer database, but it fails. Why does it fail, and what can you do to fix the problem?

Solution You need to perform a full backup before you can perform a transaction log backup. Follow these steps to perform a full backup of the ReportServer database:

1. Open Management Studio, connect to the Database Engine, expand your server, and then expand Databases.
2. Right-click ReportServer and select Properties.
3. On the Files page, verify that the Recovery model is set to Full.
4. Click OK.
5. Right-click ReportServer under Databases, point to Tasks and click Back Up.
6. In the Backup dialog box, make sure ReportServer is the selected database to back up and the Backup Type is Full.
7. Leave the default name in the Name box, and in the Description box, type **Full Backup of ReportServer**.
8. Under Destination, there may already be a disk device listed. If so, select the device and click Remove.
9. Under Destination, click Add.
10. In the Select Backup Destination box, click Backup Device, select ReportServerFull, and click OK.

11. You should now have a backup device listed under Destination. Now switch to the Options page.
12. On the Options page, select Overwrite All Existing Backup Sets. This will initialize a brand-new device or overwrite an existing one.
13. Check Verify Backup When Finished; this will check the actual database against the backup copy to see whether they match after the backup is complete.
14. Click OK to start the backup.

Manage encryption keys When the server is initialized, it acquires a symmetric key from Windows Cryptographic Services that it can use to encrypt and decrypt data. The information used to connect to the Report Server database is encrypted using the machine-level key and stored in the `RSReportServer.config` file. Stored credentials used by data sources are encrypted using the symmetric key. The unattended user account information is encrypted and stored in the `RSReportServer.config` file. Finally, the symmetric key itself is encrypted.

Master It After installing and initializing your Report Server, you realize that you need to back up the encryption keys to make sure that they are not lost. You decide to back them up to a floppy disk that you can keep off-site for added protection. How do you do this?

Solution You need to back up the encryption keys with the `rskeymgmt.exe` command=line tool:

1. Put a floppy disk in the floppy drive if you are planning on storing the file separately from the Report Server. This is optional but recommended.
2. Run `rskeymgmt.exe` on the Report Server, using the `-e` argument to extract the key, the `-f` argument to specify a location for the file on disk, and the `-p` option to protect the key with a password. Here is the command:

```
rskeymgmt -e -f a:\RSKey.snk -p password
```

Manage report processing and content One of the primary functions of Reporting Services is to store content such as reports, models, folders, resources, and shared data sources. As an administrator, you need to manage this content.

Master It One of your users requested a report through Report Manager about 10 minutes ago. The report usually only takes about 3 minutes to execute. After investigation you find that the process is stalled. What can you do to fix the problem?

Solution If a job is stuck, you can cancel it by following these steps:

1. Start Report Manager.
2. Click the Site Settings link at the top of the page.
3. Click the Manage Jobs link at the bottom of the page.
4. Check the box next to the job you want to cancel.
5. Click Cancel.

Troubleshoot common problems A number of problems can arise in Reporting Services, including issues related to HTTP, reports, processing, connections, Report Manager, subscription and delivery, initialization and encryption, and configuration.

Master It You have just installed Reporting Services on the same server as your database engine and everything seems to be working fine when you are connected to the machine locally. When you attempt to access the server over the network you get an error that reads, ''Cannot create a connection to datasource *datasourcename*.'' It is accompanied by another message that reads, ''The Report Server cannot open a connection to the Report Server database. A connection to the database is required for all requests and processing (rsReportServerDatabaseUnavailable).'' What should you do to fix the issue?

Solution This error combination means that the database server that houses the Report Server database is not configured to accept remote connections. Follow these steps to configure the SQL Server to allow remote connections:

1. Click Start ➢ Programs ➢ Microsoft SQL Server 2005 Configuration Tools ➢ SQL Server Surface Area Configuration Tool.
2. Click Surface Area Configuration For Services And Connections.
3. Open the folder for the SQL Server instance that hosts the Report Server databases.
4. Click Remote Connections.
5. Click Local And Remote Connections.
6. Click Using Both TCP/IP And Named Pipes.
7. Click OK, and then restart the service.

Chapter 11

Integrate reports into Windows and web applications.

Master It Can you bypass the Report Server's interface and request individual reports at will?

Solution The functionality of the Report Server can be harvested in two ways from within custom applications: either by requesting a report with its URL (as if it were a regular web page, including parameter values), or through the Report Server's web service, which enables you to execute code against the Report Server.

Access reports with a URL.

Master It Any report on the Report Server has its own URL, which is displayed in the browser's address box every time you click the report's link in the main page.

Solution Use the report's URL to access it as an independent web page. You can also pass parameter values to the report through its URL address. You can create shortcuts with selected report URLs on the users' desktops so that they can view the desired reports immediately without having to go through the reporting structure.

Pass commands to the Report Server through a URL.

Master It What kind of operations can you perform against Report Server from within your browser besides viewing reports?

Solution You can pass commands to the Report Server to customize the appearance of reports.

- Specify the report's output format
- Pass values for report parameters

Manage reports with the `rs` command-line utility.

Master It The Report Server comes with its own web-based management tools. Can you automate specific operations on the Report Server?

Solution The `rs.exe` command-line tool allows you to execute code against the Report Server. This utility can execute script files, which are simple text files with management commands (you can't invoke reports with a command-line tool).

Use the Report Server web service.

Master It The Report Server web service exposes a rich object model that allows you to access all of the functionality of Reporting Services from within your code. When should you use the web service?

Solution Use the methods of the Report Server web service to manage objects on the Report Server (from moving reports from one folder to another, to creating new reports from within your code) and to render reports in any output format supported by the Report Server:

- Manage report items.
- Request report in various output formats. Intercept the data transmitted by the Report Server and process them further with custom applications.
- Use the ReportExecution2005 Web service to manage the Report server's items.
- Use the ReportingService2005 web service to request and render reports.

Specify device information settings.

Master It How do you specify the settings of the device on which the report will be rendered?

Solution Supply the proper value to the `DeviceInfo` argument of the `Render` method. This argument is a short XML document, whose root element is always `<DeviceInfo>`. Insert as many elements under the root element as needed to describe the desired settings of the output.

Chapter 12

Explore the extension types Reporting Services is based on an architecture that allows you to extend its functionality by implementing custom extensions.

Master It How many types of extensions are there?

Solution

- Data processing extensions that allow you to integrate nontraditional data sources (data from flat files, XML files, and so on) to your reports
- Delivery extensions that allow you to provide your own delivery mechanisms to send report notifications to users
- Rendering extensions that allow you to create reports in totally custom formats
- Security extensions that allow you to implement custom authentication and authorization schemes

Build security extensions Security extensions are implemented as classes that expose specific functionality. This functionality is predetermined, and it's provided to you by means of interfaces: an interface is a class that contains property and method definitions, but no code. It's your responsibility to implement the members of an interface. Once an interface is implemented, you can incorporate it into Reporting Services.

Master It Which are the interfaces you must implement for a security extension?

Solution A security extension should implement two interfaces:

- The `IAuthenticationExtension` interface, which provides authentication support
- The `IAuthorizationExtension` interface, which provides authentication support

Deploy security extensions Security extensions are compiled in DLL files, which you must copy into specific folders of the Reporting Services installation. In addition, you must modify certain configuration files.

Master It Where would you copy the assembly of a security extension?

Solution The DLL(s) of a project that implements a security extension should be copied to the `ReportManager\bin` folder and the `ReportServer\bin` folder.

Build data processing extensions Data processing extensions are implemented as classes that expose specific functionality. This functionality is predetermined, and it's provided to you by means of interfaces: an interface is a class that contains property and method definitions, but no code. It's your responsibility to implement the members of an interface. Once an interface is implemented, you can incorporate it into Reporting Services.

Master It Which are the interfaces for a data processing extension?

Solution A data processing extension should implement the following interfaces:

- The `IDbConnection` interface, which represents a connection to a database server
- The `IDbCommand` interface, which represents a query or stored procedure that will be executed against the database
- The `IDbTransaction` interface, which represents a local transaction
- The `IdataParameter` interface, which represents a parameter to be passed to the query or stored procedure
- The `IDataParameterCollection` interface, which represents a collection of all parameters relevant to a command or query

- The `IdataReader` interface, which provides a method of reading a forward-only, read-only stream of data from your data source

Deploy data processing extensions Data processing extensions are compiled in DLL files, which you must copy into specific folders of the Reporting Services installation. In addition, you must modify certain configuration files.

Master It Which configuration files should you edit in the process of deploying a data processing extension?

Solution To successfully deploy a data processing extension, you must edit the `RSReportServer.config` and `rssrvpolicy.config` files. In the `RSReportServer.config` file, you must insert an <`Extensions`> tag that corresponds to your extension and has `Name` and `Type` attributes. In the `rssrvpolicy.config` file, you must add a <`CodeGroup`> tag that grants FullTrust permissions to your extension.

Chapter 13

Understand the report definition schema.

Master It To create reports from within your custom application's code, you must understand the structure of a report.

Solution Create a new XML file, apply the `ReportDefinition.xsd` schema to the new file, and then edit it in Visual Studio. The editor will suggest the available tags and attributes at each step in the IntelliSense drop-down list. Experiment with the report's elements, their values, and their hierarchy. You can also create reports with the visual tools and examine the RDL code generated by the wizard. To test your handmade report, publish it to the Report Server using the Reporting Services 2005 web service.

Define a report's data source and data sets.

Master It Most reports use a single data source and one or more `DataSets`. Data sources and `DataSets` can be specified in RDL and embedded in an RDL file, which you can then publish in a Report Server folder. How do we specify the report's data in RDL?

Solution

1. Insert a `DataSources` element under the `Report` element and specify as many `DataSource` elements as needed under it. Each `DataSource` element should have its own `ConnectionProperties` element, where you specify all the information needed to connect to the database.

2. Following the `DataSources` element, and still under the `Report` element, insert a `DataSets` element. Populate this element with as many `DataSet` elements as needed. Each `DataSet` element should have a name and a `Query` element. The `Query` element contains the command that will retrieve the desired data from the database. It also contains a `Fields` element with individual `Field` elements, where you specify the field names and map them to the database column names. To use these fields later in the document, refer to them with their names as specified in the corresponding `Field` section, and not with their names in the database.

Create report and query parameters.

Master It Most reports use parameters, and most of the report parameters are passed as arguments to the stored procedures or SQL statements that populate the `DataSet` with the report's data. You should be able to create report and query parameters from within your code.

Solution

- Report parameters are represented by `ReportParameter` elements, and they provide many properties, such as the parameter's name and type, the `AllowBlank` and `AllowNull` properties, and so on. Query parameters have only a name and a value.
- The `QueryParameter` element may appear only within a `Query` element. The `ReportParameter` element may appear only within a `Report` element.

Use the `ReportItems` element.

Master It What's a `ReportItem` and where does it appear in an RDL file?

Solution The `ReportItems` element represents the basic report entities for displaying data. A report item is usually a `Textbox` item, but it can also be a rectangle, a line, a chart, and even a subreport. The `ReportItems` element may appear in the `Body` element to specify the columns of the report, in the `Header` element to specify the column headers, and in the `Details` element to specify the values to be displayed on the report's grid.

Use the `XmlTextWriter` class.

Master It The `XmlTextWriter` class can automate the generation of XML files. Which are the basic methods of the `XmlTextWriter` class for inserting tags and attributes to an XML file?

Solution The methods are:

`WriteStartElement` Writes the opening tag for the element you pass to the method as argument

`WriteEndElement` Writes the closing tag for the most recently opened tag

`WriteAttributeString` Writes an attribute and its value as specified by the two arguments in the most recently opened element

`WriteElementString` Writes the opening and closing tags of the specified element and a value between them

Chapter 14

Explore the various types of authentication Authentication is the process of making sure that a user is who they claim to be. Authentication is provided by a combination of `ASP.NET`, Internet Information Services (IIS), and Windows security.

Master It What types of authentication are available through IIS, and which ones can you use with Reporting Services?

Solution Five types of authentication are available, but only three work with Reporting Services:

- Integrated Windows authentication, which is the default authentication method for the Report Server and Report Manager virtual directories.
- Anonymous access, which you should use only if you are using a custom security extension. Anonymous access limits your ability to vary role assignments because all users will access the Report Server under the Anonymous user account. Anonymous access also prevents users from administering the Report Server through Report Manager.
- Basic authentication, which sends usernames and passwords in clear text over the network. Consequently, you should only use this if you are also using SSL.
- Digest and Passport authentication, neither of which is supported by Reporting Services.

Learn how to set up SSL Secure Sockets Layer (SSL) provides an extra layer of security by encrypting all data that is transferred between the client and the server. It does this using a key that you can obtain from Windows Certificate Services or, better yet, from a third party such as VeriSign.

Master It You need to make sure that sensitive report data and API calls cannot be opened by malicious users while they are being transferred over the network. How can you configure the Report Server to use SSL to protect all your data?

Solution First, you need to get a certificate and apply it to the default website (or whichever website houses the Reporting Services virtual directories). Then you need to configure Reporting Services to use SSL level 3, which secures connections, report data, and API calls. Follow these steps to configure the server to use Reporting Services:

1. Assign a certificate to your default website in IIS Manager.
2. Click Start ➢ Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools ➢ Reporting Services Configuration.
3. Specify the machine name and instance name you want to connect to and click Connect.
4. Click the Report Server Virtual Directory link on the left side of the page.
5. Check the box next to Require Secure Socket Layer Connections.
6. In the Require For drop-down box, select 3-All SOAP APIs.
7. Type the fully qualified name of the server in the Certificate Name box.
8. Click Apply. A task status window appears with a report updating the system.
9. Click Exit.

Understand how to secure folders, models, and My Reports Once you have secured the server, you must secure the contents of the server. Reporting Services will ensure that users who are trying to access objects have permission to do so, a process called authorization. Reporting Services cannot do this unless you configure security for the objects you want authorization for. To start, you should secure your folders, models, and My Reports.

Master It You have created a model for the Accounting group called AcctgModel. Most of the data in the model is available to all the employees in the Accounting group, but there is one entity, named EmployeeData, that should only be available to the Accounting management staff. How can you secure the model data?

Solution Model item security is not enabled by default, so you will need to enable it using these steps:

1. In Management Studio, connect to the Report Server that stores the AcctgModel model.
2. In Object Explorer, navigate to the AcctgModel model, right-click it, and click Properties.
3. In the Model Properties page, click Model Item Security.
4. Select the Secure Individual Model Items Independently For This Model check box.
5. Select the root node. A role assignment is required on the root node.
6. Click Add Group Or User, add the group that contains all accounting employees, and click OK.
7. Navigate to the EmployeeData entity.
8. Click Add Group Or User, add the group that contains only the Accounting management staff, and click OK.

Learn how to mitigate injection attacks Injection attacks are the number-one security threat against databases. Using an injection attack, malicious users can insert their own SQL query into a text box in your application and gain access to data that they were not meant to see. Malicious users can also inject harmful scripts into web pages to accomplish similar goals.

Master It You are creating a report that uses parameters to display sensitive employee data. What should you keep in mind in order to mitigate the chance of an injection attack?

Solution One way to avoid an injection attack is to use a list of available values instead of a string parameter because a string parameter presents a text box that users can enter their own code into. If you do not show them a text box, they cannot enter their own code.

Also, you should use functions and stored procedures to return data as often as possible; it is much more difficult to inject SQL into an application that does not use hardwired SQL.

Chapter 15

Define data sources Data source definitions contain all the information needed to connect to an underlying data source, including the data source type, the connection string, and the credentials for connecting to the data source. Data sources can be report specific or shared among multiple reports.

Master It Your developers need to create several reports that all use data from the same SQL Server database. You realize that it is best to create a shared data source for all of these reports to use. How would you create the shared data source?

Solution Follow these steps to create a shared data source that points to the AdventureWorks database:

1. Open Report Manager, navigate to the folder you want to create the new data source in, and click the New Data Source button.
2. Enter a name for the data source in the Name text box.
3. Optionally, enter a description in the Description text box that will appear on the Contents page in Report Manager.
4. Select Microsoft SQL Server from the Connection Type list.
5. In the Connection String text box, enter this connection string to connect to the AdventureWorks database:

```
data source=local; initial catalog=AdventureWorks
```

6. Select the Windows Integrated Security option under Connect Using.
7. Click OK.

Define data sets You can create one data set for each data source in your report. A data set contains such data as which fields you will use from the data source, calculated fields you may create, a pointer to the data source, parameters, and data options like case sensitivity and collation settings.

Master It You need to be able to use SSIS packages to create data sets for some of your reports. What must you do to enable this functionality?

Solution To enable the data processing extension for the Report Designer, follow these steps:

1. Install Integration Services on the same computer on which you installed Reporting Services.
2. Using Windows Explorer, locate <*drive*>:\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies.
3. Using a text editor, open the RSReportDesigner.config file.
4. Remove the comment marks (delete <!-- and --> characters from the beginning and end of the line) for this entry in the <Data> section:

```
<!-- <Extension Name="SSIS"
   Type="Microsoft.SqlServer.Dts.DtsClient.DtsConnection, ➢
Microsoft.SqlServer.Dts.DtsClient, Version=9.0.242.0, ➢
Culture=neutral, PublicKeyToken=89845dcd8080cc91"/> -->
```

5. In the <Designer> section, remove the comments from this entry:

```
<!-- <Extension Name="SSIS"
   Type="Microsoft.ReportingServices.QueryDesigners. ➢
GenericQueryDesigner,Microsoft.ReportingServices. ➢
QueryDesigners"/>  -->
```

6. Save the file.

Then follow these steps to configure the extension for the report server:

1. Open the `RSReportServer.config` file, located by default at *X*`:\Program Files\Microsoft SQL Server\MSSQL.`*n*`\Reporting Services\ReportServer`.
2. Remove the comment marks (delete <!-- and --> characters from the beginning and end of the line) for this entry:

```
<!-- <Extension Name="SSIS" ➢
Type="Microsoft.SqlServer.Dts.DtsClient.DtsConnection, ➢
Microsoft.SqlServer.Dts.DtsClient, Version=9.0.242.0, ➢
 Culture=neutral, PublicKeyToken=89845dcd8080cc91"/> -->
```

3. Save the file.

Master It You are creating a new report that uses the FirstName, LastName, and SSN columns from the Employee table in your HumanResources database. The database is on a SQL Server 2005 system. How would you create the data set?

Solution First, you need to create a data source, and then you can create the data set. Here are the steps to create the data source and data set:

1. Create a new report in Business Intelligence Development Studio.
2. On the Data tab select <New Dataset> from the Dataset drop-down list.
3. In the Dataset dialog box, select New Data Source from the Data source drop-down list, or select a shared data source if one is available.
4. In the Data Source dialog box, from the Type drop-down list select Microsoft SQL Server.
5. Specify a connection string that works with your SQL Server data source. This example would connect you to the AdventureWorks database on the local server:

```
Data Source=localhost;Initial Catalog=HumanResources;
```

6. Set the database credentials in the Data Source dialog box on the Credentials tab.
7. Click OK.
8. Enter this query in the Query string text box to populate the data set:

```
SELECT FirstName, LastName, SSN FROM HumanResources.Employee
```

9. Click OK.

Chapter 16

Use performance counters Using Performance Console, you can monitor objects and counters. An object is a part of the system, such as the processor or the physical memory. A counter displays the statistical information about how much that object is being used. Reporting Services installs performance counters for both the Windows and web services.

Master It Some of your users have been complaining that the Report Server is not returning reports in a timely manner, and some of the users are getting error messages.

You want to find out how many requests for reports are being processed by the web service, how many of those requests are successful, and how many fail. What counters should you monitor?

Solution Using the MSRS 2005 Web Service object, you can monitor these counters:

- Total Requests — The total number of reports that ran successfully after the service started
- Total Processing Failures — The total number of report processing failures that have occurred since the service started
- Total Reports Executed — The total number of all requests made to the Report Server after the service started, whether or not they are successful

Specify settings in each of the configuration files All of the settings that Reporting Services requires to run are stored in configuration files saved on the hard disk. The configuration files all contain XML data, so you can use any XML editor or text editor to modify the files, though some of the settings are encrypted, which means you must use Reporting Services tools to modify them.

Master It What are the primary configuration files, where are they stored, and what is each one used for?

Solution There are four primary configuration files:

- `RSReportServer.config` holds Report Server web service and Windows service configuration data. This is stored in the `\Reporting Services\ReportServer` directory.
- `RSWebApplication.config` contains Report Manager configuration data. This is stored in the `\Reporting Services\ReportManager` directory.
- `ReportingServicesService.exe.config` holds Report Server Windows service configuration data that is specifically used to configure trace settings. This is saved in the `\Reporting Services\Report Server\Bin` directory.
- `RSReportDesigner.config` contains Report Designer configuration settings. This is saved in the `\Program Files\Microsoft Visual Studio 8\Common7\IDE\PrivateAssemblies` directory.

Monitor and configure Reporting Services Before you can monitor Reporting Services to see if it is running slower than normal, you need to know what normal is. Set a measurement baseline to define normal. You should then continuously monitor Reporting Services to make sure it is running as expected and keep the logs for trend tracking and archiving. If the system is running slow, you may have to adjust the amount of available memory.

Master It Some of your users are complaining that they are receiving an out-of-memory exception when they try to use the Report Server. You have 4GB of RAM, so you know that the system is not actually running out of memory. The users who get this error are all using service provided by the web service. What is the problem and what can you do to fix it?

Solution The problem is that your system is not configured to use 4GB of memory by default. Remember that `ASP.NET` only uses a maximum of 2GB of memory by default, but the `memoryLimit` setting in the `machine.config` file specifies that it is allowed to use 60 percent of the available memory, which is 2.4GB. To fix the issue, you can lower the `memoryLimit` value in the `machine.config` file or add the `/3GB` setting in your `boot.ini` file.

Appendix B: SharePoint Integration

While I was still writing this book, Microsoft SQL Server 2005 Service Pack 2 was released to production. That release has a big change that affects Reporting Services; you can now integrate Reporting Services with Microsoft Windows SharePoint Services 3.0 and Microsoft Office SharePoint Server 2007 (which I will refer to collectively as SharePoint).

In a nutshell, SharePoint is collaboration software. SharePoint allows you to create a website (or several websites) that users can access to share documents, complete surveys, locate information on colleagues, and so on. SharePoint stores all of the dynamic information that users create in SQL Server databases, just as Reporting Services stores report data in SQL Server databases. So, if you are running SharePoint and Reporting Services, you can integrate the two to get added functionality. There are just a few requirements you must meet to integrate:

- You must use SQL Server 2005 Reporting Services with Service Pack 2. You cannot integrate a Report Server instance that is running earlier versions of the software.
- You must be running Evaluation, Standard, Developer, or Enterprise editions. You cannot integrate SharePoint with Workgroup Edition or Express Edition with Advanced Services.
- You must run either Windows SharePoint Services 3.0 or Microsoft Office SharePoint Server 2007. If you are using Office SharePoint Server, you must have the Office Server Premium or the Office Server Standard Edition.
- You also need to download and install the Reporting Services Add-in for SharePoint Technologies, which is freely available from Microsoft.

Once integrated, Reporting Services provides several new features that are only available when running in integrated mode:

- You can use SharePoint's document management and collaboration features, including alerts and version control, on reports. A SharePoint site provides a unified portal for accessing and managing all documents in one place.
- You can secure reports, models, and other items using SharePoint permissions and authentication providers.
- You can use SharePoint deployment topologies to distribute reports over an Internet connection outside the firewall.
- You can create custom SharePoint application pages to manage reports, models, data sources, schedules, and report history. You can also set properties, define schedules and subscriptions, and create and manage report history on a SharePoint site the same way you create and manage them from other tools in SQL Server.
- Users can view a report in a new customizable Report Viewer web part, which includes page navigation, search, print, and export features.

- You can publish or upload reports, report models, resources, and shared data source files to a SharePoint library, including Report Center in Office SharePoint Server.
- You can generate models and start Report Builder from a SharePoint site, and you can add, open, and save ad hoc reports to a SharePoint site.
- You can create subscriptions that deliver reports using the new SharePoint delivery extension.
- If you are using Office SharePoint Server, you can connect a Report Viewer web part to a Filter web part or a Library web part in dashboards or web part pages.
- Developers can program against a new SOAP endpoint to create custom applications that integrate with a SharePoint site. Service Pack 2 also offers an updated Windows Management Instrumentation (WMI) provider that you can use to programmatically configure a Report Server instance that runs in SharePoint integrated mode.

Unfortunately, some Reporting Services features work differently, or not at all when running in integrated mode:

- In SharePoint integrated mode, SharePoint URLs are used to reference reports, report models, shared data sources, and resources instead of the Report Server folder hierarchy.
- You cannot use, or even deploy, Reporting Services custom security extensions on the Report Server. This is because the Report server includes a special-purpose security extension that is used whenever you configure a Report Server to run in SharePoint integrated mode. This security extension is an internal component, and it is required for integrated operations.
- You will not be able to use Report Manager or Management Studio to manage a Report server instance that is configured for SharePoint integration.
- Data-driven subscriptions are not available in any edition.
- The `rs.exe` command-line utility is not supported in integrated mode. This is because the utility does not support the SOAP endpoint used for programmatic access to a Report Server that runs in SharePoint integrated mode.
- Linked reports are not supported.
- My Reports is not supported.
- Job management features that allow you to stop a long-running report process are not supported.
- Batching methods are not supported.

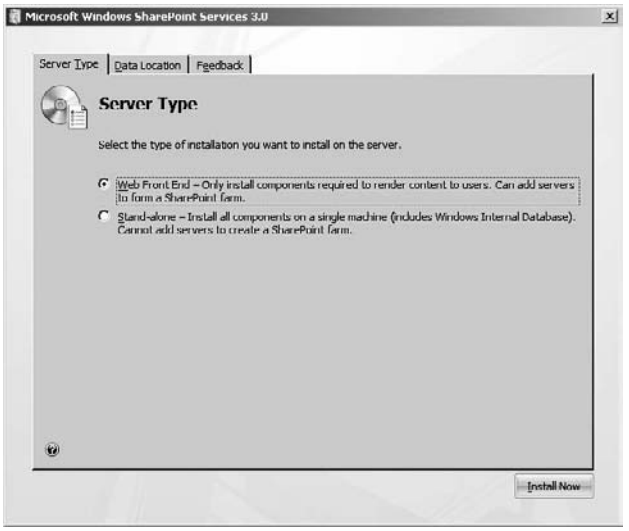
We have discussed SharePoint integration briefly at different points in the book, so in this appendix, I will give you more detail on how to configure Reporting Services and Windows SharePoint Services 3.0 for integration, and I will show you how to perform your day-to-day tasks after the two systems are integrated.

Integrating with SharePoint

Before you can do anything else with the two systems, you need to install Windows SharePoint Services, create a SharePoint website, and configure Reporting Services for SharePoint integration. Let's start by downloading, installing, and configuring Windows SharePoint Services 3.0 (WSS):

In this exercise we are going to assume that you are installing SharePoint on a server that already has Reporting Services installed.

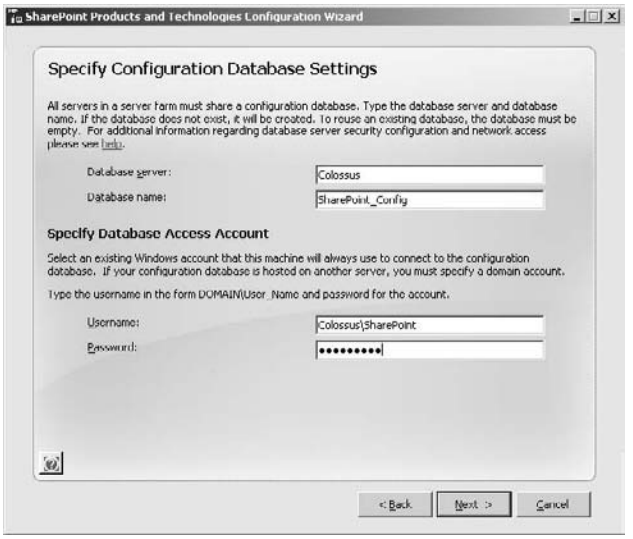
1. WSS 3.0 requires the .NET Framework 3.0. Download and install it from `http://www.microsoft.com/downloads/details.aspx?familyid=10CC340B-F857-4A14-83F5-25634C3BF043&displaylang=en`.
2. Download `SharePoint.exe` and save it to disk. You can get it from here: `http://www.microsoft.com/downloads/details.aspx?familyid=D51730B5-48FC-4CA2-B454-8DC2CAF93951&displaylang=en`.
3. Run `SharePoint.exe` from the location you saved it to on disk.
4. On the first screen, accept the license agreement and click Continue.
5. On the Choose The Installation You Want screen, click Advanced.
6. On the Server Type screen, check Web Front End.



7. Click Install Now.
8. When installation is complete, click Close.
9. Now the SharePoint Configuration Wizard opens. On the Welcome screen, click Next.
10. A dialog box pops up asking you if it is acceptable to stop IIS and two SharePoint services. Click Yes.

11. On the Connect To A Server Farm screen, select the No, I Want To Create A New Server Farm option and click Next.

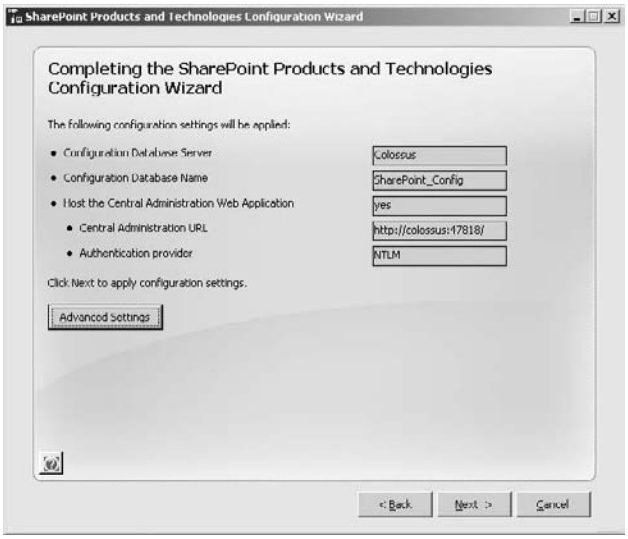
12. On the Specify Configuration Database Settings page, enter the name of your database server in the Database Server text box, and then enter a username and password for a user that will be used to create and connect to the SharePoint databases. The account you use here should have System Administrator permissions on SQL Server.



13. Click Next.

14. Accept the default settings on the Configure SharePoint Central Administration Web Application page and click Next.

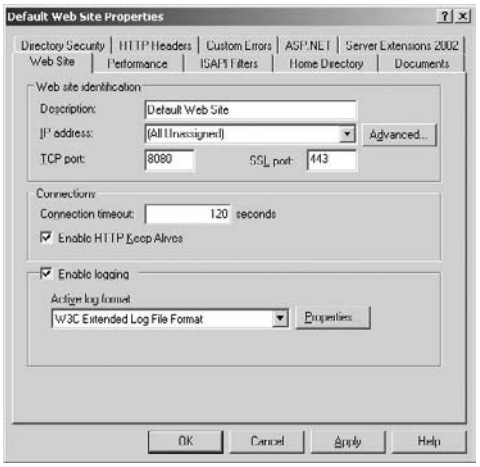
15. On the Completing The SharePoint Products And Technologies Configuration Wizard screen, verify your settings and click Next.



16. When the configuration is complete, click Finish.

Because you installed WSS on a system that already has Reporting Services installed, you will need to change the port number of the Default Web Site in IIS; otherwise the Reporting Services Web service will stop working. Here's how to do it:

1. Open Internet Information Services (IIS) Manager from Administrative Tools on the Start menu.
2. Expand your server, then Web Sites, then right-click the Default Web Site and select Properties.
3. Enter an unused port number in the TCP Port text box (8080 is a good example).



4. Click OK.
5. Next, verify that you can still access the Report Server by typing a URL that includes the port. If you used port 8080, you should be able to access the Report Server at `http://server:8080/reportserver`.
6. Now you need to edit the `RSReportServer.config` file to add the port to the `UrlRoot` configuration setting. Open `RSReportServer.config` from *X:*`\Program Files\Microsoft SQL Server\MSSQL.`*N*`\Reporting Services\ReportServer`.
7. Find the `URLRoot` setting and add **:8080** after the server name in the URL. It should look like this: `http://server:8080/reportserver`.

SharePoint and Reporting Services are both installed on the same server now, but they are not integrated. Before you can link the two, you must install SQL Server 2005 Service Pack 2. Here's how:

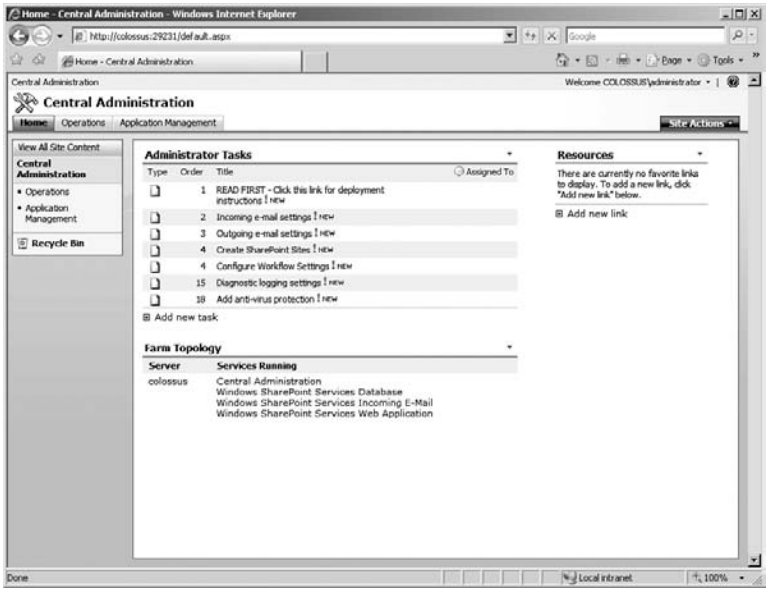
1. Download SQL Server 2005 Service Pack 2 and save it to disk. You can get it here: `http://www.microsoft.com/downloads/details.aspx?FamilyId=d07219b2-1e23-49c8-8f0c-63fa18f26d3a&DisplayLang=en`.
2. Run `SQLServer2005SP2-KB92186-`*xxx-xxx*`.exe` from the location where you saved it on disk. The last six characters differ from file to file; they represent the processor

architecture and language. For example, `SQLServer2005SP2-KB92186-x86-ENU.exe` is the 32-bit, English version.

3. On the Welcome screen, click Next.
4. Accept the license agreement and click Next.
5. Accept the defaults on the Feature Selection screen and click Next.
6. Accept the defaults on the Authentication screen and click Next.
7. Accept the defaults on the Error And Usage Reporting Settings screen and click Next.
8. If you are presented with a Running Processes screen, you can stop all of the processes listed to avoid a reboot, or you can just click Next and reboot the server after installation is complete. Either way, click Next when you are ready.
9. On the Ready To Install screen, click Install.
10. On the Installation Complete page, click Next.
11. For this exercise, we will assume that you are running Windows Server 2003, so uncheck the box Launch The User Provisioning Tool For Windows Vista after SP2 Installation Completes. Then click Finish.
12. Reboot your system if necessary.

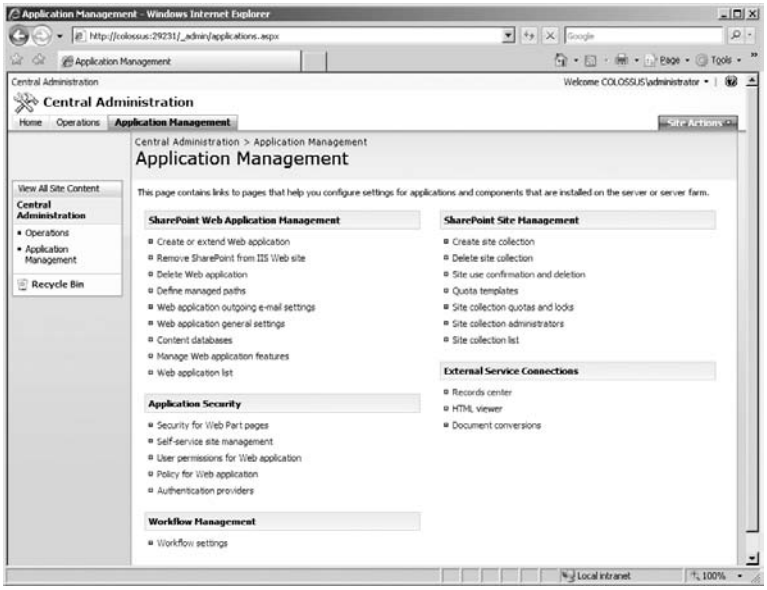
Now that you have Service Pack 2 installed, you need to create a SharePoint website:

1. Open SharePoint 3.0 Central Administration from Administrative Tools on the Start menu.

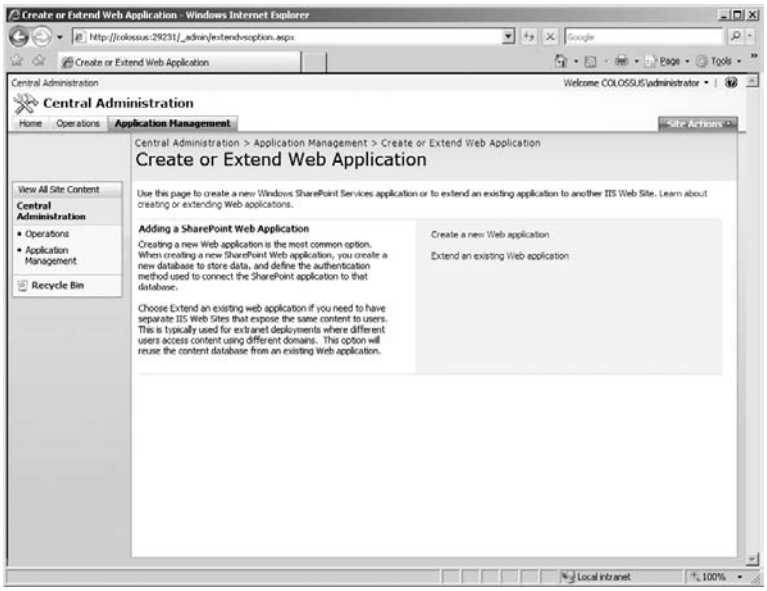


2. Click the Application Management link on the left side of the page.

3. In the SharePoint Web Application Management section, click the Create Or Extend Web Application link.

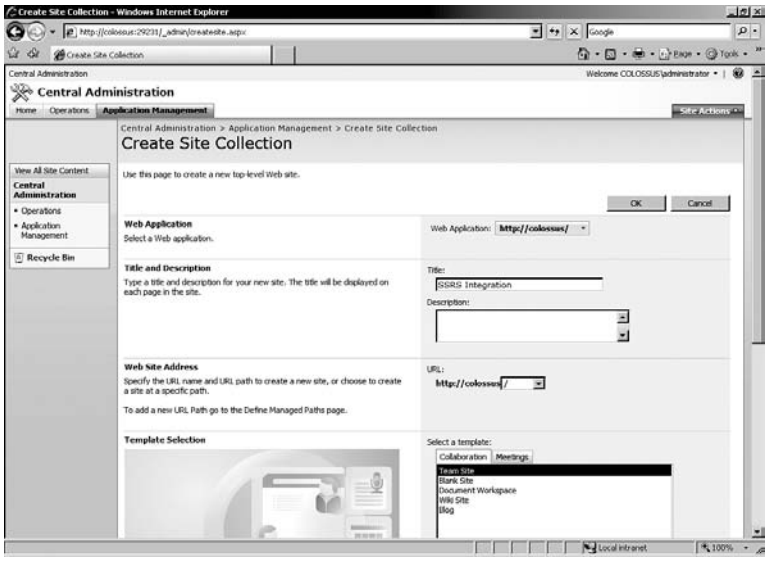


4. On the page that comes up, click the Create A New Web Application link.

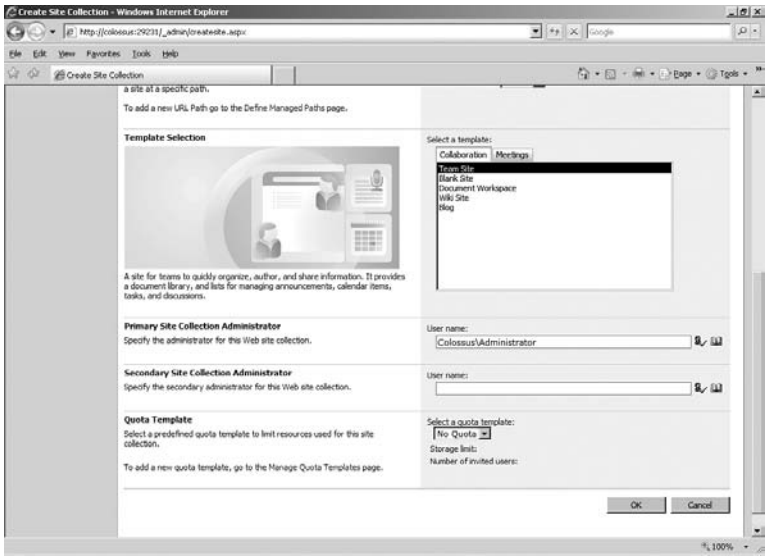


5. On the Create New Web Application page, scroll down to the Application Pool section and select the Use Existing Application Pool option.

6. Click OK at the bottom of the page (or the top of the page — they both do the same thing).
7. After the application has been created, click Start, select Run, enter **iisreset /noforce**, and click OK.
8. Go back to the Application Management page and click the Create Site Collection link in the SharePoint Site Management section.
9. In the Title text box, type **SSRS Integration**.



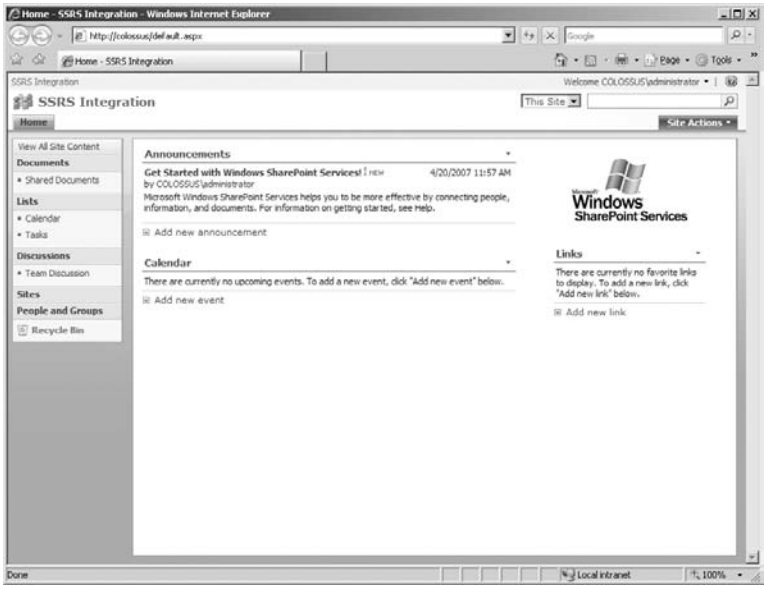
10. Scroll down to the Primary Site Collection Administrator section and enter your username in the text box in the *domain\user* format.



11. Click OK.

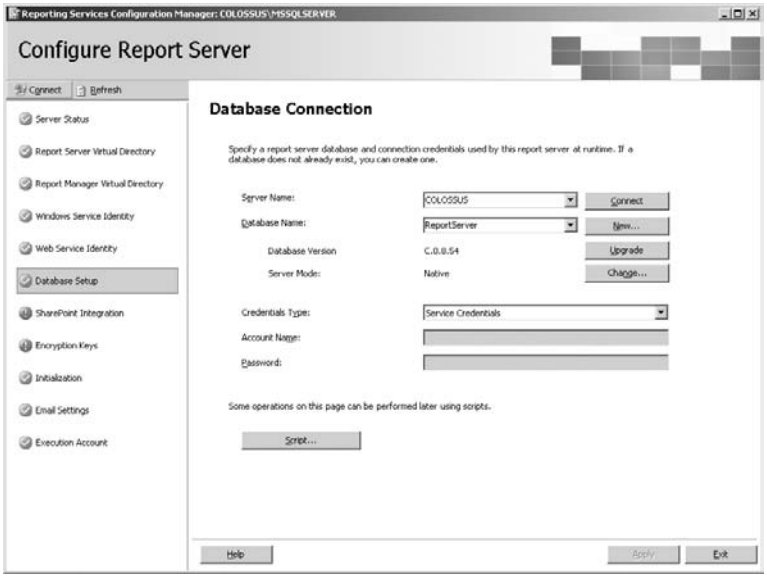
12. After the site has been created, click OK.

13. Open a new browser window and navigate to the new SharePoint site using this URL: `http://localhost`.

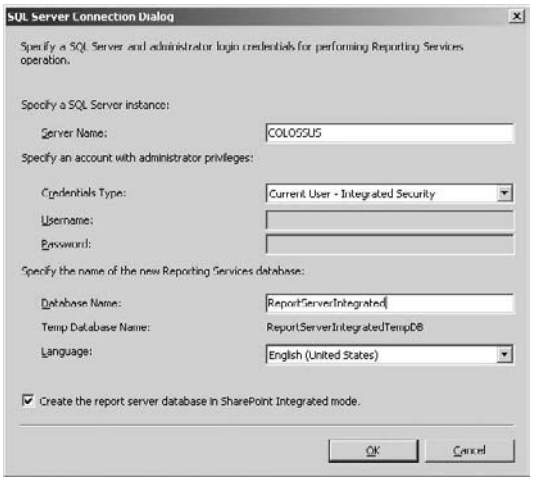


Now that you have a working SharePoint website, you are ready to integrate Reporting Services with SharePoint. Here s how to do it:

1. Select Start ➢ Programs ➢ Microsoft SQL Server 2005 ➢ Configuration Tools. Then open Reporting Services Configuration and connect to your Report Server.
2. Click Database Setup to open the Database Setup page.
3. Click the Change button.



4. A dialog box will warn you that you need to create a new database to continue with integration. Click Yes.
5. In the SQL Server Connection dialog box, enter **ReportServerIntegrated** in the Database Name text box. Notice that the Create The Report Server Database In SharePoint Integrated Mode option is checked for you.



6. Click OK.

7. After the database has been created, click Apply, and then in the subsequent dialog box, click Connect.

8. Close the Reporting Services Configuration tool.

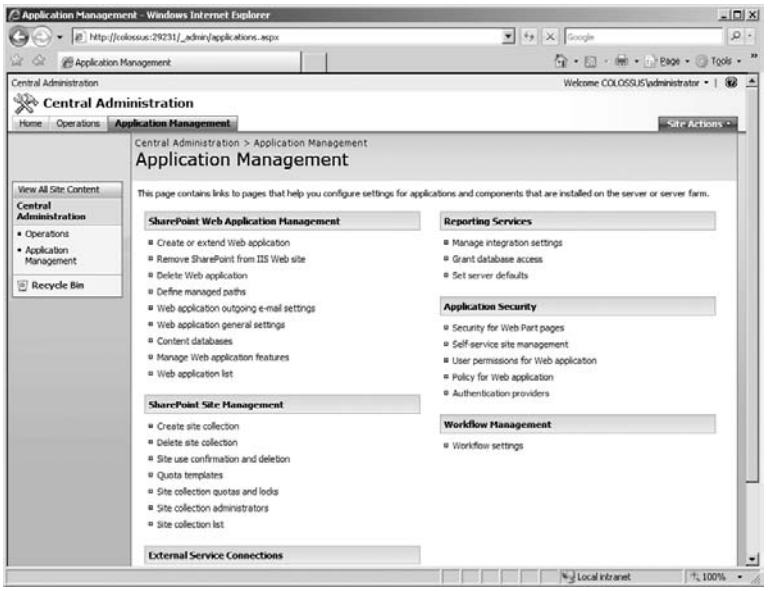
Now you are ready to install the Reporting Services Add-in, which is a freely available download from Microsoft. Let's do that now:

1. Download the Reporting Services Add-in and save it to disk. You can get it from here: `http://www.microsoft.com/downloads/details.aspx?FamilyId=1e53f882-0c16-4847-b331-132274ae8c84&DisplayLang=en`.

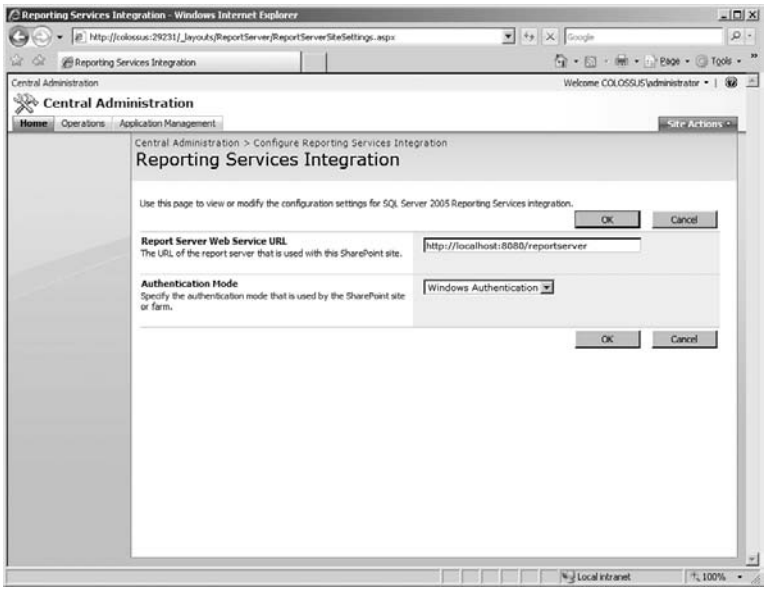
2. Run `SharePointRS.msi` from the location you saved it on disk.

Now you need to specify the integration settings in the SharePoint server configuration:

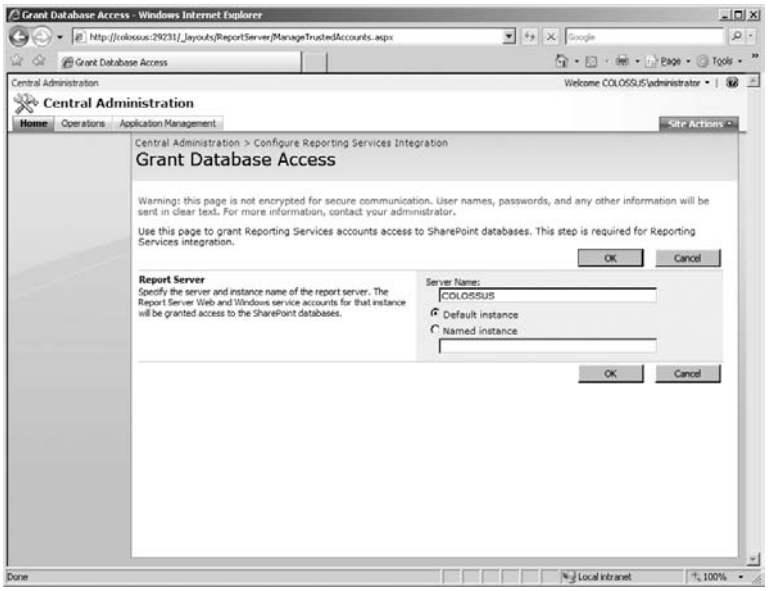
1. Open SharePoint 3.0 Central Administration from Administrative Tools and switch to the Application Management page. You should see a new section named Reporting Services.



2. In the Reporting Services section, click Manage Integration Settings.
3. In Report Server Web Service URL, enter **`http://localhost:8080/reportserver`**.
4. In Authentication mode, select Windows Authentication. You could also select Trusted Authentication to specify whether the URL proxy endpoint sends a header with a security token for an impersonated connection on the Report Server, but do not do that here.



5. Click OK to apply the changes and go back to the Application Management page.
6. Now you need to grant the Report Server account access to the SharePoint databases. So, in the Reporting Services section, click Grant Database Access.
7. On the Grant Database Access screen, specify the name of the Report Server. By default, this is the name of the computer.
8. Specify whether the instance is the default instance or a named instance. If you are not sure, start the Reporting Services Configuration tool and connect to the Report Server. In the Instance Selection dialog box, expand Instance Name to view a list of the instances installed on the server. MSSQLSERVER is the name of the default instance. All other instances are named instances.



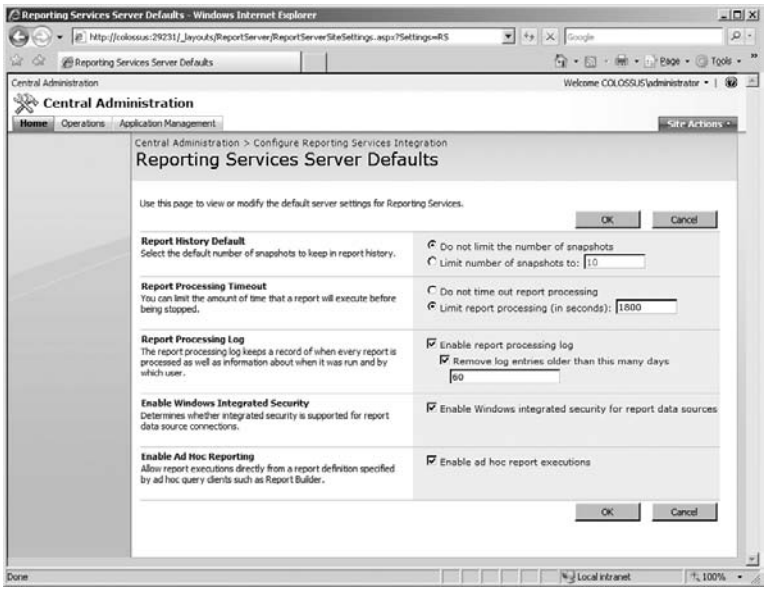
9. Click OK.

10. In the Enter Credentials dialog box that opens, specify the username and password of a local administrator (your own account should work), and then click OK.

11. Back at the Application Management page, in the Reporting Services section, click Set Server Defaults. You can leave the default values for these settings or change them if you like:

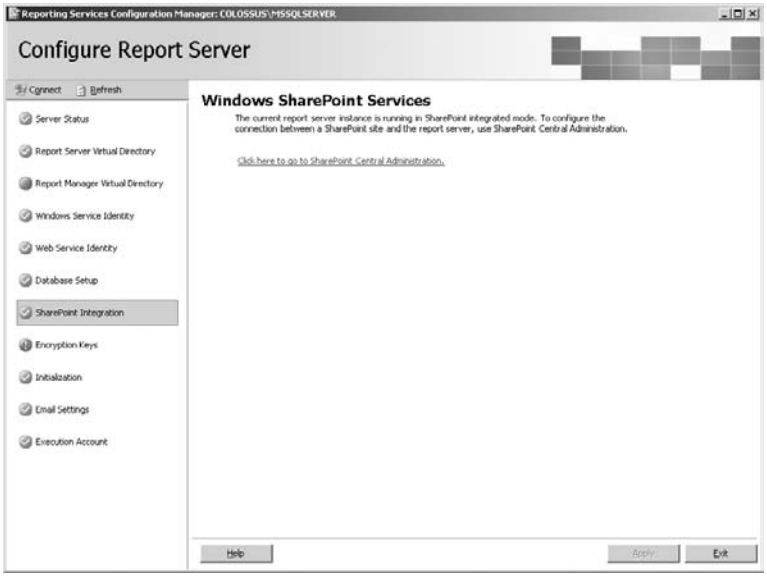
    - The Report History Default setting specifies a default value for the number of copies of report history to retain for the entire site. You can override this value by specifying different limits in property pages for specific reports.
    - Set the Report Processing Timeout to specify whether report processing times out on the Report Server after a certain number of seconds.
    - In Report Processing Log, you can specify whether the Report Server generates trace logs and the number of days the log is kept. The logs are stored on the Report Server computer in the `\Microsoft SQL Server\MSSQL.N\ReportServer\Log` folder. A new log file is started each time the service is restarted.
    - In Enable Windows Integrated Security section, you can specify whether or not users can access a report data source using their existing Windows security token.

- In Enable Ad Hoc Reporting section, specify whether users can perform ad hoc queries from a Report Builder report. If this option is deselected, the Report Server will not generate click-through reports for any report that uses a report model as a data source.



12. Click OK.

13. To verify that all of this worked as expected, open the Reporting Services Configuration tool. The SharePoint Integration icon should be green. If you switch to the SharePoint Integration page, you will get a link to the SharePoint 3.0 Central Administration page. Please note that you may need to stop and restart the server on the Server Status page for the icon to change to green.

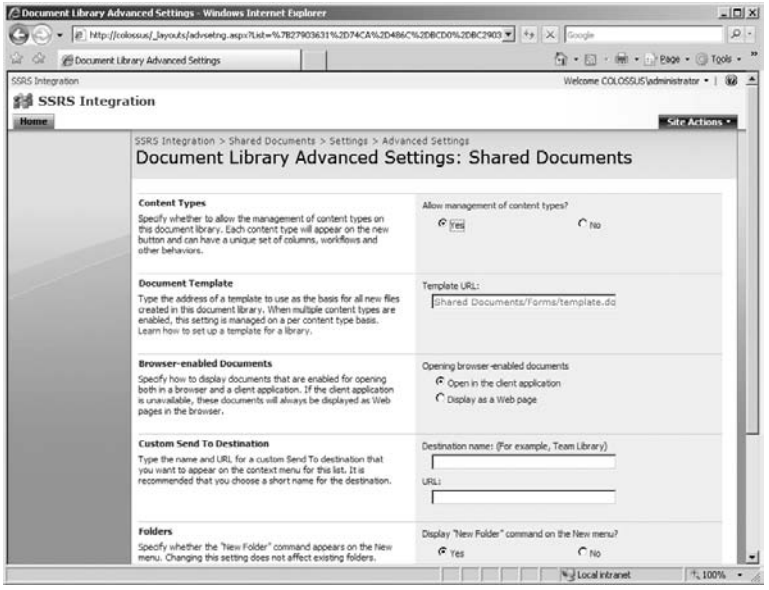


There is one final task before you start using the Report server in integrated mode. Reporting Services provides several content types that are used to create and work with reports:

- `.rsds` files are for shared data sources.
- `.smdl` files are for report models.
- `.rdl` files are for reports.

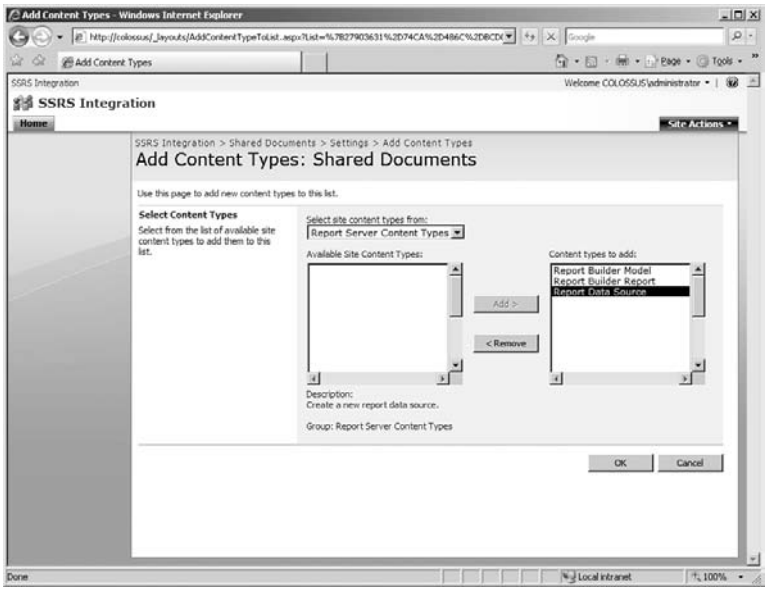
To use these content types with SharePoint, you need to register them with the library you will be using. To do this, you must be an administrator or have Full Control permissions on the library. First, you need to enable content type management for the library:

1. Open your SharePoint site in a web browser and click the Shared Documents link on the left side of the page.
2. On the Settings menu, click Document Library Settings.
3. Under General Settings, click Advanced settings.
4. In the Content Types section, select Yes to allow multiple content types.
5. Click OK.



Now you can add Report Server content types to the library:

1. On the Customize Shared Documents page, under Content Types, click Add From Existing Site Content Types.
2. Select Report Server Content Types from the Select Site Content Types From drop-down list.
3. In the Available Site Content Types list, click Report Builder Model, and then click Add to move the selected content type to the Content Types To Add list.
4. Add Report Model and Report Data Source content types by selecting each one and clicking the Add button.



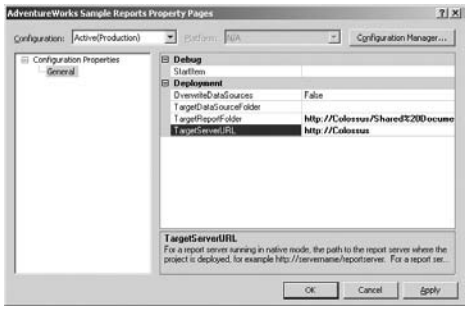
5. When you finish selecting all of the content types that you want to add, click OK.

Now the two servers are integrated and you can start working with them. Let's see how to accomplish some of the everyday tasks that you need to perform in this new mode.

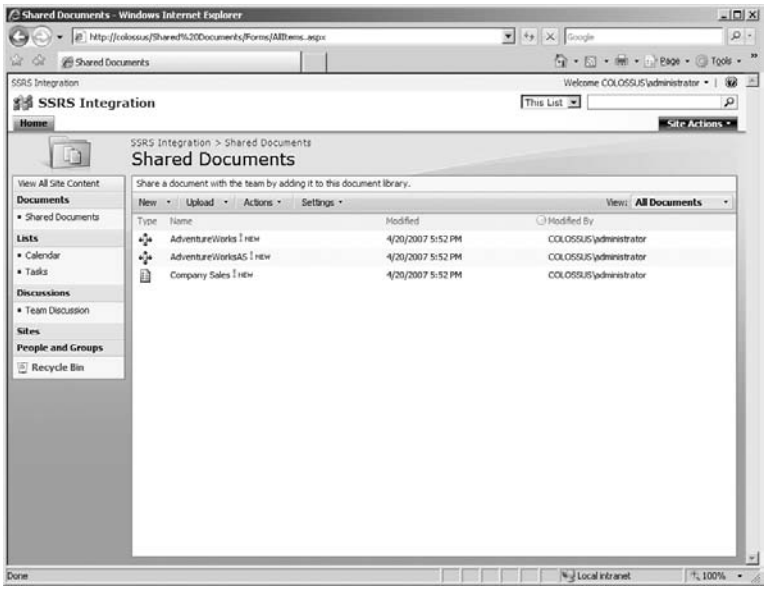
Publishing Reports

One of the first things you will want to do after integrating Reporting Services with SharePoint is create and publish a new report. You can use Business Intelligence Development Studio (BIDS) to create a new report and publish it to a SharePoint library. Let's publish one of the AdventureWorks sample reports to the Shared Documents library:

1. Open Business Intelligence Development Studio and open the AdventureWorks Sample Reports project.
2. From the Project menu, click AdventureWorks Sample Reports Properties.
3. For TargetReportFolder, type a URL to a library or library folder — for example, `http://servername/Shared%20Documents`.
4. For TargetServerURL, type the URL for the SharePoint top-level site: `http://servername`.



5. Click OK.
6. In the Solution Explorer, right-click the Company Sales report and click Deploy.
7. Open your SharePoint website in a browser and click the Shared Documents link. You should now see the Company Sales report and the corresponding data sources.



To view the report, you can click the link in the document library, which will bring up the report in the Report Viewer.

Working with Subscriptions

When running in integrated mode, you can still create subscriptions to reports stored in a SharePoint library. The report must use stored credentials and the user creating the subscription needs to have permission to view the report and create SharePoint alerts. Four delivery methods are available:

- File share
- E-mail
- Null delivery
- Document library

The file share, e-mail delivery, and null delivery methods work the same in integrated mode as they do in native mode. The document library method is specific to integrated mode, though,

so you need to know a few things about it. The document library delivery processor creates a static file that is not actually a report, so it cannot be run in the Report Viewer web part and it has no interactive features. Also, you must deliver the report to a document library on the same site as the report; you cannot deliver the report to a library on another server or on another site on the same server.

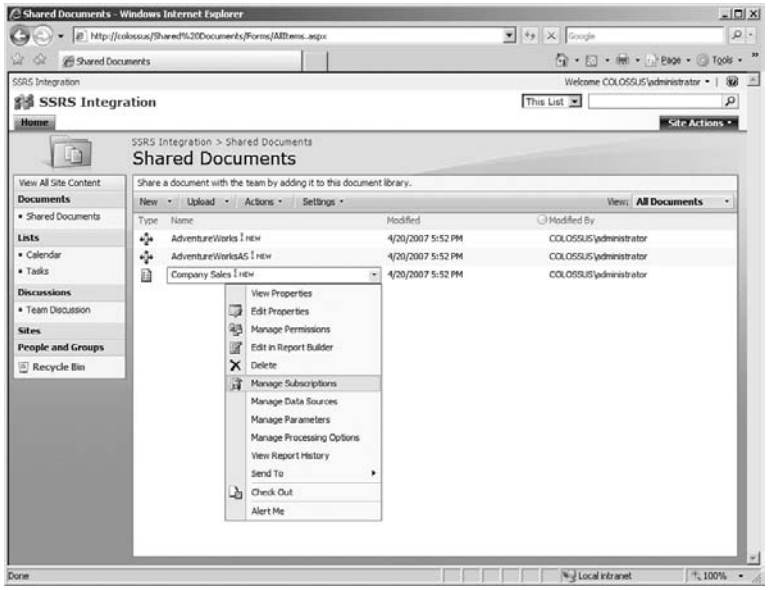
Before you can create a subscription, you must provide credentials for the data source. Let's configure the AdventureWorks data source now:

1. Open the Shared Documents library in your SharePoint website.
2. Click the AdventureWorks link.
3. In the Credentials section, select the Stored Credentials option.
4. Type your own username and password in the appropriate boxes.
5. Check the Use As Windows Credentials option.

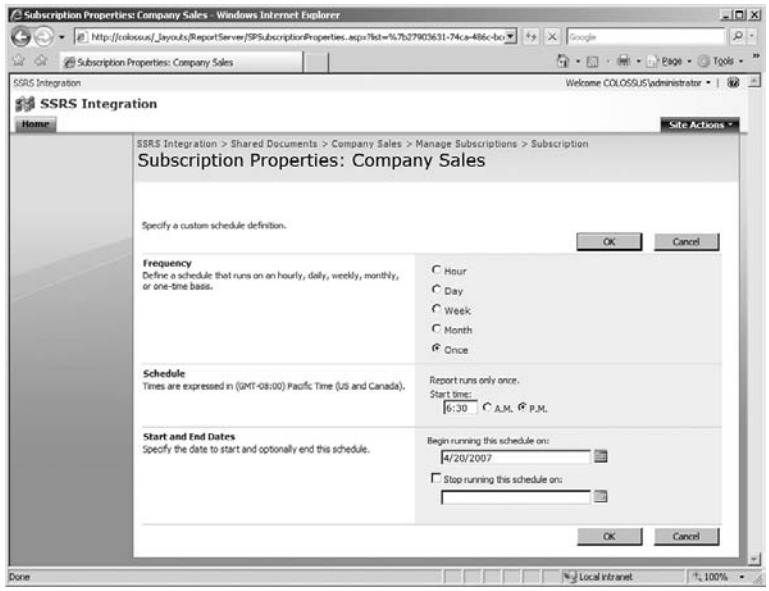
6. Click OK.

Now you can create a subscription to the Company Sales report. Let's create a subscription that delivers this report to a SharePoint document library:

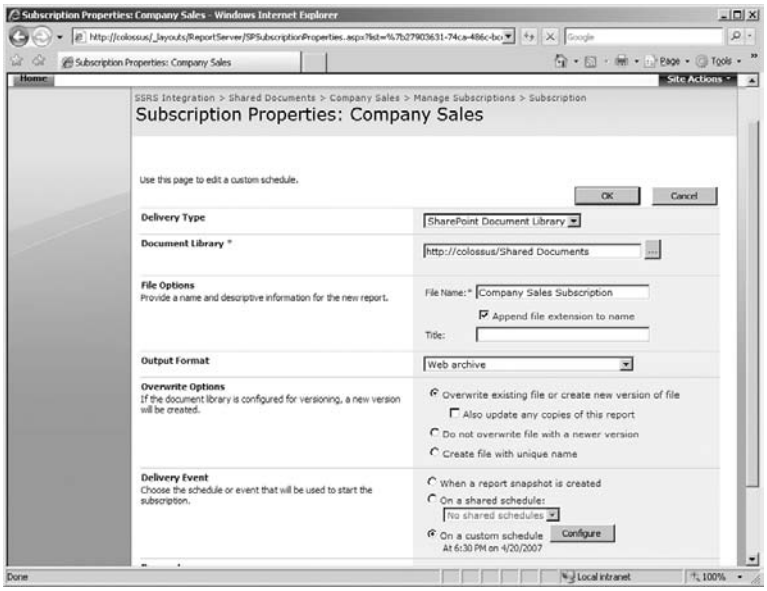
1. Open the Shared Documents library in your SharePoint website.
2. Hover over the Company Sales report.
3. Click the down arrow next to the report and select Manage Subscriptions.



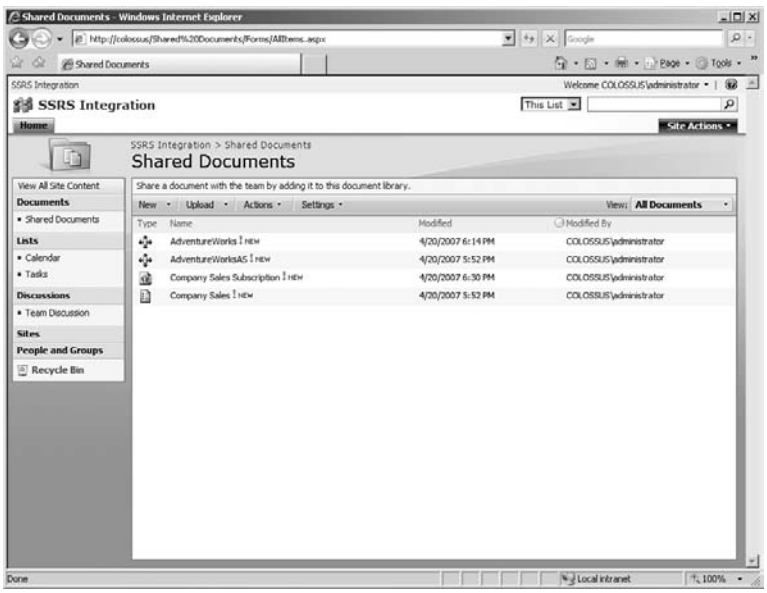
4. Click Add Subscription.
5. In Delivery type, select SharePoint Document Library.
6. In Document Library, select Shared Documents. This should already be filled in for you.
7. In File Options, change the filename to **Company Sales Subscription**.
8. In Output Format, select Web Archive.
9. In Delivery Event, click the Configure button to create a schedule.
10. Select Once for the frequency and enter a time that is 10 minutes from now for the Start time.



11. Click OK to return to the Subscription Properties page.
12. Click OK to create the subscription.



13. Wait until the start time you entered in step 10. After that time you should see a new document named Company Sales Subscription in the Shared Documents library.



Of course, subscriptions can outlive their usefulness, so you will need to delete them from time to time. In these steps you will delete the subscription you've just created:

1. Open the Shared Documents library in your SharePoint website.
2. Hover over the Company Sales report.

3. Click the down arrow next to the report and select Manage Subscriptions.
4. Check the box next to the subscription you created in the last set of steps.
5. Click Delete and then click OK in the confirmation dialog box.

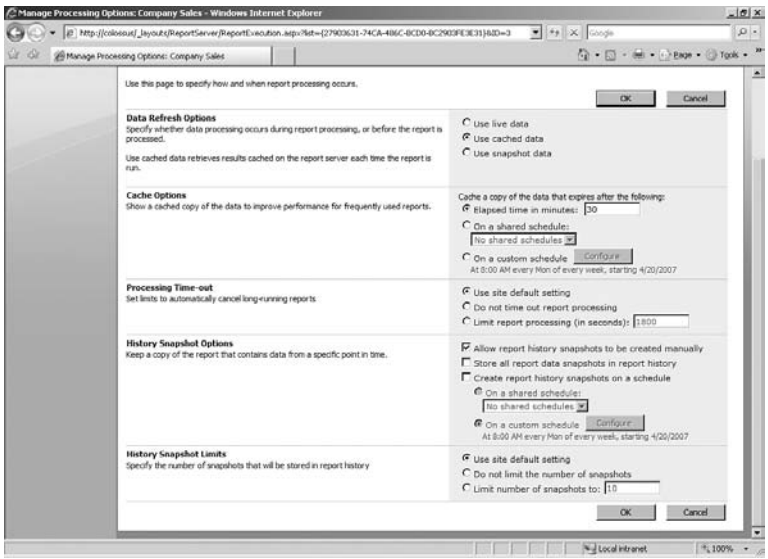
Setting Processing Options

There are several processing options that you can set for the reports in your SharePoint libraries. For instance, you can specify whether the report is run from live data, from a cached copy of the report, or from a snapshot. You can also set a timeout so the report does not take excessive resources on the server. There is also an option for limiting the number of snapshots that will be stored in the report history.

You can instruct the server to keep a report history, which is a collection of previously run reports. This is valuable when you need to keep track of a report as it is run over time. You should not keep a report history for sensitive or confidential reports.

In this series of steps, you will configure the data refresh processing options for the Company Sales report:

1. Open the Shared Documents library in your SharePoint website.
2. Hover over the Company Sales report.
3. Click the down arrow next to the report and select Manage Processing Options.
4. In the Data Refresh Options section, select the Use Cached Data option.
5. In the Cache Options section, leave the default value of 30 minutes for the Elapsed Time In Minutes option.



6. Leave the remaining default values and click OK.

Using the Report Viewer Web Part

The concept of *web parts* was introduced in Windows SharePoint Services 2.0 and has since been incorporated into ASP.NET 2.0 and Windows SharePoint Services 3.0. Web parts allow users to personalize their web pages so that they contain content that is relevant to them. For example,

one user could add a web part that contains their contacts; another user could add a web part that shows the weather. Neither of these users would see the web parts that the other user has added to their personal version of the page.

As an administrator, you can add web parts to the shared page, which everyone who visits that site can see. This is helpful if you want everyone in the company to have access to certain data at all times. For example, you may want to add a company announcements web part to the shared page; then everyone in the company can see the important announcements when they view the shared page.

One of the web parts that you and your users can add is the Report Viewer web part. This web part allows you to view reports that are stored in a SharePoint library. Users need to have the Add And Customize Pages SharePoint permission to add this web part.

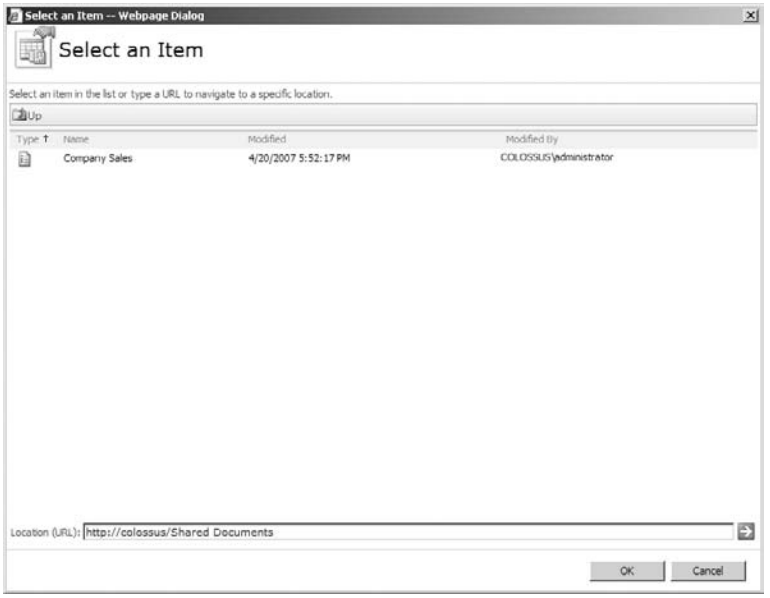
Even though they have the same name, the Report Viewer web part that is installed with the Reporting Services Add-in is different from the Report Viewer web part that is installed with the `RSWebParts.cab` file. Use the version that is installed with the Reporting Services add-in when running in integrated mode.

Next, you will add the web part to a web page and configure it to display a report:

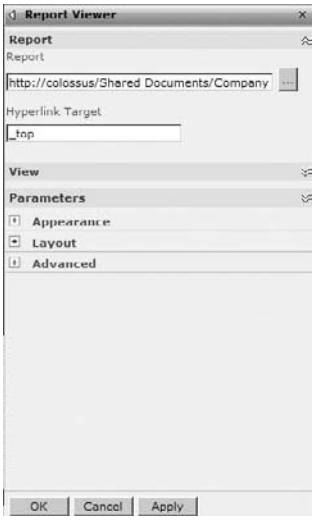
1. Open your SharePoint website.
2. On Site Actions, click Edit Page.
3. Click Add A Web Part in the Left web part zone.
4. In All Web Parts, in the Miscellaneous section, check the box next to SQL Server Reporting Services Report Viewer.

5. Click Add. The web part is added at the top of the zone, but you can drag it to a different location in the zone.
6. At the top of the web part, click the Edit button and select Modify Shared Web Part to open the tool pane.

7. Click the ellipsis (. . .) button next to the Report text box.
8. Click the Shared Documents link on the Select An Item page.
9. Select the Company Sales report and click OK.



10. Click Apply at the bottom of the tool pane, and then click OK to close the pane.



11. The Company Sales report should now be displayed in the Report Viewer web part. Click the Exit Edit Mode link at the top right of the page (just below Site Actions) to return to the normal page view.

Appendix C: Report Server Web Service Reference

Reporting Services exposes its functionality through two web services, which you can use from within your code to do anything you can do with Report Manager. In effect, Report Manager uses the two web services to interact with the Report Server. The two web services of Reporting Services are:

ReportExecution2005 Web Service Allows developers to programmatically process and render reports from a Report Server

ReportService2005 Web Service Allows developers to programmatically manage objects in a Report Server

In this appendix I discuss the methods and complex types exposed by both web services. Due to space limitations, I can't provide an example for each and every method, so I will demonstrate with sample code only the most common and difficult methods. To experiment with the members of the two services, you can either write short Windows applications that call their methods, or you can write an RSS script and execute it with the `rs.exe` utility. If you choose to write a Windows application, you must add to the project a reference to the appropriate web service. Right-click the project's name in the Solution Explorer and select the Add Web Reference command from the context menu. In the Add Reference dialog box, click the link Web Services on the Local Machine. When the list of web services on the local machine appears, locate either the ReportService2005 or the ReportExecution2005 web service, enter a name for the reference in the Web Reference Name box, select the component in the left pane, and click the Add Reference button to add the reference to your project. Use the **ReportingService2005** and **ReportExecution2005** web services. Then declare a variable to represent each reference in your code:

```
Dim RS As New ReportService.ReportingService2005
Dim RE As New ReportExecution.ReportExecution2005
```

For the code samples, I'm using the reference name `RS` for the Report Service web service and the name `RE` for the Report Execution web service. To use these two variables, you must first authenticate the client application. There are two ways to authenticate your application: Windows authentication (the application passes to the web service the credential of the user who is executing the application) and Basic authentication (the application passes to the web service the username and password of an account). The first method is simpler and safer, because you

don't have to store passwords anywhere; you simply execute the following statement, which uses the current user's credentials to connect to the Report Server:

```
RS.Credentials = System.Net.CredentialCache.DefaultCredentials
```

or

```
RE.Credentials = System.Net.CredentialCache.DefaultCredentials
```

To use the Basic authentication method, you must create a new instance of the `System.Net.NetworkCredential` class, passing the current user's name and password in the instance's constructor, as in the following statement:

```
RS.Credentials = New System.Net.NetworkCredential(➢
              "username", "password", "domain")
```

Just set the three parameters of the method to the username, password, and name of the domain controller that will validate the credentials. If you want to experiment with the ReportService2005 web service using the `rs.exe` utility, create a text file with the script and save it with the `.rss` extension (you can't call the methods of the ReportExecution2005 web service from within a script). Then open a command prompt window and execute the following command:

```
rs -i script_file.rss -s http://localhost/ReportServer
```

where `script_file.rss` is the text file with the script you want to execute. The script file should contain a public subroutine called `Main()`, and you must insert the code you want to execute in this subroutine. You don't have to declare the `RS` variable in the script; this step is taken care of for you. However, you must supply the proper credentials. Here's a sample script for exercising the `ListChildren` method of the ReportService2005 web service:

```
Public Sub Main()
   rs.Credentials = System.Net.CredentialCache.DefaultCredentials
   Dim items() As CatalogItem
   items = rs.ListChildren("/", False)
   Dim item As CatalogItem
   For Each item In Items
      Console.WriteLine(item.Name)
   Next
End Sub
```

The Warning Class

A very common class in both web services is the `Warning` class, which is used with many of the methods described in this appendix, so it's presented before the members of the two web services. Many methods return the list of errors or warnings generated by Report Server as an array of `Warning` objects, one such object per warning or error message. You can read the descriptions

returned by Report Server from within your code and act accordingly. Here are the properties of the `Warning` class:

`Code` A read-only property that reads the error code assigned to the warning by the Report Server.

`Message` A read-only property that reads the error description for the warning.

`ObjectName` A read-only property that reads the name of the object that's responsible for the warning.

`ObjectType` A read-only property that reads the type of the object that caused the error or warning.

`Severity` A read-only property that reads the severity type of the error or warning.

The `CreateReport` method, for example, returns an array of `Warning` objects. The following code iterates through the elements of the `Warnings` array and displays them in the output window for debugging purposes (I'm not showing the arguments of the `CreateReport` method; this sample shows you how to use the `warnings` array; see the description of the `CreateReport` method for more information on creating reports in code):

```
warnings = rs.CreateReport(. . .)
If Not (warnings Is Nothing) Then
    Dim warning As Warning
    For Each warning In  warnings
        Console.WriteLine(warning.Message)
    Next warning
Else
    Console.WriteLine("Operation completed w/o warnings")
End If
```

ReportingExecution2005 Web Service

The ReportingExecution2005 web service provides the methods you need to execute your reports. It exposes two properties: ExecutionHeaderValue and ServerInfoHeaderValue.

ExecutionHeaderValue

Returns information about the current session and its state. The `ExecutionHeader` object, among other properties inherited from the `SoapHeader` class, exposes the `ExecutionID` property, which returns a unique ID that identifies the current session.

ServerInfoHeaderValue

Returns information about the Report Server. The `ServerInfoHeader` class, among other properties inherited from the `SoapHeader` class, exposes the following properties:

`ReportServerDateTime` Returns the current date and time of the Report Server

`ReportServerEdition` Returns the edition of the Report Server as a string: `Developer`, `Evaluation`, `Enterprise`, or `Standard`

ReportServerVersion Returns the version of the Report Server installation as a string.

ReportServerVersionNumber Returns the version number of the Report Server installation as a string.

ReportExecution Methods

The ReportExecution2005 web service provides the following methods for executing reports and interacting with them from within your code.

FindString

Returns the logical page number of the next occurrence of the specified string:

```
FindString (StartPage As Integer, _
      EndPage As Integer, FindValue As String) As Integer
```

The `StartPage` parameter is the number of the first page to be searched and `EndPage` is the last page of the search. Set both parameters to 0 to search the entire report. The `FindValue` parameter is the string to search for. The method returns a numeric value, which is the number of the page that contains the next occurrence of the string (or 0 if the string is not found in the report's body).

GetDocumentMap

The `GetDocumentMap` method accepts no arguments and returns the representation of the document map for the current report as a `DocumentMapNode` object.

The DocumentMapNode Class

The `DocumentMapNode` class provides the following properties:

Children Gets or sets a collection of child document map nodes

Label Gets or sets the label to be displayed for this node

UniqueName Gets or sets the unique name of the report item or grouping that this node should point to

GetExecutionInfo

The `GetExecutionInfo` method accepts no arguments and returns information about the current report as an `ExecutionInfo` object. The ExecutionInfo class provides the following properties.

AllowQueryExecution A true/false value indicating whether the user is allowed to provide values for parameters used in the query.

AutoRefreshInterval Represents the rate, in seconds, at which a report page rendered as HTML automatically refreshes.

CredentialsRequired A true/false value indicating whether the report needs to be supplied with credentials.

DataSourcePrompts The prompts for all data sources associated with a report. This is an array of `DataSourcePrompt` objects.

ExecutionDateTime The date and time that the snapshot associated with the execution was created.

ExecutionID A string that uniquely identifies the current execution.

ExpirationDateTime The report's expiration date and time.

HasDocumentMap A true/false value indicating whether the report has a document map.

HasSnapshot A true/false value indicating whether data has been retrieved and processed for the report.

HistoryID The history ID of the report, if it comes from a report history snapshot.

NeedsProcessing A true/false value indicating whether the snapshot associated with the execution needs to be created or reprocessed.

NumPages The number of logical pages, including soft page breaks, in the report.

Parameters Contains parameter information for the report execution.

ParametersRequired A true/false value indicating whether the report requires parameter values to be supplied.

ReportPageSettings Represents the report page settings for the current execution.

ReportPath The path to the report on the server.

DataSourcePrompt

Represents the name of a data source and the prompt to display to a user; consists of these properties:

DataSourceID The unique ID of a data source

Name The data source's name

Prompt The prompt to display when prompting the user for data source credentials

GetRenderResource

Returns the resource for the specified rendering extension format.

```
GetRenderResource (Format As String, ➢
   DeviceInfo As String, ➢
   ByRef MimeType As String) As Byte()
```

The first parameter is the format of the rendering extension for which you want to retrieve the resource, and it's a string like `PDF` or `CSV`. To find out the supported extensions, call the `ListRenderingExtensions` method. The `DeviceInfo` parameter represents device-specific settings for the specified rendering extension, and the last parameter is the resource's MIME type. The rendering extension resource is returned as a Base 64-encoded byte array.

ListRenderingExtensions

Returns a list of rendering extensions available on the Report Server's installation. The method returns an array of `Extension` objects.

ListSecureMethods

Returns the names if the methods that require a secure (SSL) connection to be invoked as an array of strings.

LoadDrillthroughTarget

Creates the subreport corresponding to a drillthrough action as an `ExecutionInfo` object. This method accepts as an argument the ID of the item that's the drillthrough target and returns its result as an `ExecutionInfo` object.

LoadReport

Loads a report from the Report Server catalog for execution and should be called before the `Render` method:

```
LoadReport (Report As String, ➢
            HistoryID As String) As ExecutionInfo
```

The `Report` parameter is the report's name and the `HistoryID` parameter is the ID of the history snapshot from which the report will be rendered. To use live data, set this parameter to `Nothing`. The method returns an `ExecutionInfo` object.

Logoff

Logs out the current session. The application must log on again by calling the `LogonUser` method.

LogonUser

Logs a user on to the Report Server web service:

```
LogonUser (userName As String, ➢
           password As String, authority As String)
```

The first two parameters are the user's Windows account name and password. The third argument, which is optional, is the domain that will authenticate the user. Use the `Nothing` value to omit this parameter.

NavigateBookmark

Navigates to a specific bookmark in the report:

```
NavigateBookmark (BookmarkID As String, ➢
                  ByRef UniqueName As String) As Integer
```

The first parameter is the ID of the `DocumentMapNode` that represents the bookmark, and the second parameter is the name of the report item that the client should use to position the view area. The second parameter is passed by reference, because the method may set the item to which it will position the view area at the client. The method returns the number of the page to which the bookmark belongs, or 0 if the specified bookmark is not found.

NavigateDocumentMap

Navigates to a specific section of the report:

```
NavigateDocumentMap (DocMapID As String) As Integer
```

`DocMapID` is the ID of the `DocumentMapNode` object of the desired section. The method returns the number of the page that contains the specified section, or 0 if the specified section is not found.

Render

Renders a report in a specific format:

```
Render (Format As String, DeviceInfo As String, ➢
        ByRef Extension As String, ByRef MimeType As String, ➢
        ByRef Encoding As String, ByRef Warnings As Warning(), ➢
        ByRef StreamIds As String()) As Byte
```

The `Format` parameter is a string identifying the format in which the report will be rendered (it's a rendering extension's name). Report Server supports the following extensions: `XML`, `NULL`, `CSV`, `IMAGE`, `PDF`, `HTML4.0`, `HTML3.2`, `MHTML`, `EXCEL`, and `HTMLOWC`. You can retrieve the list of supported extensions with the `ListRenderingExtensions` method. The `DeviceInfo` parameter is an XML segment with device-specific content. The `Extension` parameter is the file extension for the specified format, and `MimeType` is the MIME type of the rendered report. The `Encoding` parameter is set to the encoding of the report's contents. The last two arguments are arrays: `Warnings` is an array of `Warning` objects with all warning issued by the Report Server while processing the report, and `StreamIDs` is an array of stream identifiers. Each external resource (such as images) and individual pages of a multipage report can be retrieved by passing its ID to the `RenderStream` method. The `Render` method does not return a value.

The following statements render a report and save the output of the `Render` method as a web page. If the report has parameters, you should specify them with the `SetExecutionParameters` method, discussed a little later in this appendix. You must also call the `LoadReport` method before rendering a report.

```
Dim out as Byte()
Dim warnings() As Warning
rs.LoadReport("/AWCustomers/YearlySales", Nothing)
out = rs.Render("MHTML", Nothing, Nothing, ➢
               Nothing, Nothing, warnings, Nothing)
Dim strm As FileStream = ➢
      File.Create("CustomerReport.mht", out.Length)
strm.Write(out, 0, out.Length)
strm.Close()
```

When would you use multiple streams to retrieve the rendered report? Let's say you have a long report that you want to retrieve as a series of images. To specify that the report is to be rendered as images, you must pass the following device information to the `Render` method with

the DeviceInfo argument (it's a long string and you can type it on a single line, but I broke it in multiple lines for readability):

```
deviceInfo = ➢
"<DeviceInfo>" & ➢
"   <OutputFormat>JPEG</OutputFormat>" & ➢
"   <PageWidth>13in</PageWidth>" & ➢
"</DeviceInfo>"
```

The deviceInfo setting specified the output format and the page size. This will force the Report Server to generate multiple images, one per report page. Also, you must pass the string "IMAGE" with the Format argument and the "JPEG" specification (or "GIF" or "PNG" depending on the image format) with the Extension argument. This is how you must call the Render method to retrieve multiple streams, one for each image:

```
Dim result As Byte()
Dim StreamIDs As String() = Nothing
result = RE.Render("IMAGE", deviceInfo, "JPG", ➢
                   Nothing, "en-us", warnings, StreamIDs)
```

At this point we can find out the number of images (it's the number of elements in the StreamIDs array), and call the Render method once for each page of the report, passing the page number through the DeviceInfo argument:

```
Dim pageCount As Integer = StreamIDs.Length
Dim img As Image
If pageCount > 1 Then
    For i As Integer = 1 To pageCount
      deviceInfo = String.Format(➢
               "<DeviceInfo>" & ➢
               "<OutputFormat>JPEG</OutputFormat>" & ➢
               "<StartPage>{0}</StartPage>" & ➢
               "<PageWidth>13in</PageWidth>" & ➢
               "</DeviceInfo>", i)
      result = RE.Render("IMAGE", deviceInfo, Nothing, ➢
                         Nothing, Nothing, Nothing, StreamIDs)
      img = System.Drawing.Image.FromStream(➢
                  New System.IO.MemoryStream(result))
' Process of display the img Image object
    Next
End If
```

Each call to the Render method now returns a different page of the report, rendered as a JPEG image. The image's bytes are stored in the result array, which it then converted to an Image object. You can insert code to save the images, display them on a PictureBox control of a Windows application, or make a web page and transmit to a client.

RenderStream

Renders a stream in the specified format:

```
RenderStream (Format As String, StreamID As String, ➢
              DeviceInfo As String, ByRef Encoding As String, ➢
              ByRef MimeType As String) As Byte()
```

The `Format` parameter specifies the format in which to render the stream. This argument maps to a rendering extension. Supported extensions include `XML`, `NULL`, `CSV`, `IMAGE`, `PDF`, `HTML4.0`, `HTML3.2`, `MHTML`, `EXCEL`, and `HTMLOWC`. A list of supported extensions may be obtained by calling the `ListRenderingExtensions` method. `StreamID` is a stream identifier. The `DeviceInfo`, `Encoding`, and `MimeType` parameters are the same as with the `Render` method. The method returns the stream as an array of bytes.

ResetExecution

Resets the current report execution.

SetExecutionCredentials

Sets the credentials associated with the current report execution:

```
SetExecutionCredentials (Credentials As DataSourceCredentials())➢
                         As ExecutionInfo
```

The `Credentials` parameter is an array of `DataSourceCredentials` objects and the method returns an `ExecutionInfo` object. If a report uses multiple data sources, you must specify the credentials for all data sources at once.

SetExecutionParameters

Sets and validates the parameter values associated with a report:

```
SetExecutionParameters (Parameters As ParameterValue(),➢
                        ParameterLanguage As String) ➢
                        As ExecutionInfo
```

The `Parameters` parameter is an array of `ParameterValue` objects, which correspond to the report's parameters, and the `ParameterLanguage` parameter is the locale identifier that should be used to interpret the values (values like dates, for example, depend on the locale). Use a few statements like the following to set the report's parameters:

```
Dim parameters(1) As ParameterValue
parameters(0) = New ParameterValue()
parameters(0).Name = "Year"
parameters(0).Value = "2007"
```

```
parameters(1) = New ParameterValue()
parameters(1).Name = "CustomerID"
parameters(1).Value = "318"
rs.LoadReport("/AWCustomers/YearlySales", Nothing)
rs.SetExecutionParameters(parameters, "en-us")
```

The ReportParameter Class

The `ReportParameter` class represents a report parameter and exposes the following properties:

`AllowBlank` A read-only property that indicates whether the current parameter may accept an empty string as a value.

`AllowBlankSpecified` Indicates whether the `AllowBlank` property for the current parameter has been specified.

`DefaultValues` Gets or sets the default values of the current parameter.

`DefaultValuesQueryBased` A read-only true/false value indicating whether the default values of the parameter are based on a query.

`DefaultValuesQueryBasedSpecified` A true/false value indicating whether the `DefaultValuesQueryBased` property for the current parameter has been specified.

`Dependencies` Gets or sets a list of parameters that will be used to retrieve parameter values with a query.

`ErrorMessage` The error message to return when the current parameter's validation fails.

`MultiValue` A true/false value that indicates whether the current parameter is a multivalued one.

`MultiValueSpecified` A true/false value indicating whether the `MultiValue` property has been specified.

`Name` Reads or sets the current report parameter's name.

`Nullable` A true/false value indicating whether the value of the current parameter can be null.

`NullableSpecified` A true/false value indicating whether the `Nullable` property has been specified.

`Prompt` The text that prompts the user to provide the current parameter's value.

`PromptUser` A true/false value indicating whether the user is prompted for the value of the current parameter.

`PromptUserSpecified` A true/false value indicating whether a value for the `PromptUser` property is specified.

`QueryParameter` A true/false value indicating whether the parameter is used in a query.

`QueryParameterSpecified` A true/false value indicating whether the `ReportParameter` property has been specified.

`State` Returns the state of the current parameter; its value is a member of the `ParameterStateEnum` enumeration: `DynamicValuesUnavailable`, which means the parameter value is unavailable; this state indicates that query-based values are invalid;

`HasOutstandingDependencies`, which means the parameter has outstanding dependencies; this generally occurs when a set of query based values hasn't been supplied; `HasValidValue`, which means a valid value exists for the parameter; or `MissingValidValue`, which means there are no valid values for the parameter.

`StateSpecified` A true/false value indicating whether the `State` property has been specified.

`Type` The data type of the parameter. Its value is a member of the `ParameterTypeEnum` enumeration: `Boolean`, `DateTime`, `Float`, `Integer`, and `String`.

`TypeSpecified` A true/false value indicating whether the `Type` property has been specified.

`ValidValues` A read-only property that holds the parameter's valid values.

`ValidValuesQueryBased` A true/false read-only value indicating whether the parameter's valid values are based on a query.

`ValidValuesQueryBasedSpecified` A true/false value indicating whether the `ValidValuesQueryBased` property has been specified.

Sort

Applies or removes a sort for the report based on a user sort action:

```
Sort (SortItem As String, Direction As SortDirectionEnum, ➢
      Clear As Boolean, ByRef ReportItem As String, ➢
      ByRef NumPages As Integer) As Integer
```

The first parameter is the ID of the item on which the rows will be sorted, and the `Direction` parameter, which specifies the direction for the sort, can have one of the following values: `Ascending`, `Descending`, or `None`. The `Clear` parameter should be set to true to clear any existing sorts. Its default value is false, so that you can apply multiple sorts. Finally, the `ReportItem` parameter is the ID of the item that will be used to position the report in the viewing area, and `NumPages` is the number of pages in the report after the sort (its value is set by the `Sort` method). The method returns the number of the page that contains the item specified by the `ReportItem` parameter.

ToggleItem

Toggles the show/hide property of a report item:

```
ToggleItem (ToggleID As String)
```

This method accepts as argument the ID of an item and toggles its `Visible` property. The method returns true, unless the item is not found.

ReportingService2005 Web Service

The ReportingService2005 web service provides methods and properties that can be used to call the Microsoft SQL Server 2005 Reporting Services (SSRS) web service. This web service exposes the functionality for managing resources on the Report Server. You can use its methods to set up schedules and subscriptions, and even create new reports in code. However, you can't render reports with this web service; for this purpose you must use the `ReportExecution2005` class.

ReportService Methods

The ReportService2005 web service provides the following methods for manipulating the objects (such as reports, schedules, resources, and data sources) on the report server from within your code.

CancelBatch

The `CancelBatch` method accepts no arguments and cancels the batch that was initiated by a call to the `CreateBatch` method.

CancelJob

Cancels the execution of a job:

```
CancelJob(JobID) As Boolean
```

This method accepts as argument the ID of the job to be cancelled and returns a true/false value indicating whether the job was cancelled successfully.

CreateBatch

Creates a batch with multiple methods that will be executed in the context of a single transaction by the `ExecuteBatch` method. The method accepts no parameters and returns a string that uniquely identifies the batch. Operations that depend on the outcome of previous operations, such as removing a folder after successfully removing all items in the folder, should be executed in the context of a transaction: if one operation fails, then the previous operations should be rolled back and any subsequent operations should be aborted. The actions specified in a batch operation are not executed until you call the `ExecuteBatch` method, which requires the identifier created by the `CreateBatch` method.

Use this method to create a new batch, assign it to the `BatchHeaderValue` property of a variable that represents the ReportingService2005 web service, and then specify the commands of the batch:

```
Dim bh1 As New BatchHeader()
Dim bh2 As New BatchHeader()
' Create batch ...
bh1.BatchID = rs.CreateBatch()
rs.BatchHeaderValue = bh1
rs.CreateFolder("Customers", "/AWReports", Nothing)
rs.CreateFolder("Suppliers", "/AWReports", Nothing)
rs.CreateFolder("Creditors", "/AWReports", Nothing)
' and execute it
rs.ExecuteBatch()
' Create a second batch ...
bh2.BatchID = rs.CreateBatch()
rs.BatchHeaderValue = bh2
rs.DeleteItem("/AWReports/Customers")
rs.DeleteItem("/AWReports/Suppliers")
rs.DeleteItem("/AWReports/Creditors")
' and execute 2nd batch
rs.ExecuteBatch()
```

CreateDataDrivenSubscription

Creates a data-driven subscription for the specified report:

```
CreateDataDrivenSubscription (➢
     Report As String, ExtensionSettings As ExtensionSettings, ➢
     DataRetrievalPlan As DataRetrievalPlan, ➢
     Description As String, EventType As String, ➢
     MatchData As String, ➢
     Parameters As ParameterValueOrFieldReference()) ➢
     As String
```

The `Report` parameter is the full pathname of the report for which a new data-driven subscription will be created. The second parameter, `ExtensionSettings`, is an `ExtensionSettings` object that contains the settings of a specific delivery extension. The `DataRetrievalPlan` parameter is an object that provides settings required to retrieve data from a delivery query. `Description` is the job's description (the text displayed to the users), and `EventType` is the type of event that triggers the data-driven subscription. The valid values are `TimedSubscription` or `SnapshotUpdated`. The `MatchData` argument contains data that is associated with the specified `EventType` parameter. Finally, the `Parameters` parameter is an array of `ParameterValueOrFieldReference` objects that contains the report's parameters. The method returns a string with a unique identifier for the data-driven subscription.

CreateDataSource

Creates a new data source in the Report Server catalog:

```
CreateDataSource ➢
   (DataSource As String, Parent As String, ➢
    Overwrite As Boolean, Definition As DataSourceDefinition, ➢
    Properties As Property())
```

`DataSource` is the name of the data source and `Parent` is the full pathname of the folder containing the data source. The `Overwrite` parameter is a true/false value indicating whether an existing data source with the same name should be overwritten. The `Definition` parameter is a `DataSourceDefinition` object that describes the connection properties for the data source. Finally, the `Properties` parameter is an array of `Property` objects with name/value pairs that sets the properties of the data source.

The DataSourceDefinition Class

The `DataSourceDefinition` class represents the attributes of a data source and provides the following properties:

`ConnectString` The data source's connection string.

`CredentialRetrieval` Indicates the way in which the Report Server retrieves data source credentials. This property's value is a member of the `CredentialRetrievalEnum` enumeration: `None` (no credentials required — this is the least secure method of connecting to a data source); `Prompt` (the Report Server prompts the user for credentials); `Integrated` (the Report Server uses the credentials of the authenticated user); and `Store` (the credentials are stored securely in the Report Server catalog).

Enabled A true/false value indicating whether a data source is enabled.

EnabledSpecified True if the Enabled property has been specified.

Extension The data source extension: SQL, OLEDB, ODBC, or a custom extension.

ImpersonateUser A true/false value indicating whether the Report Server attempts to impersonate a user by using stored credentials after a data processing extension has established an authenticated connection to a data source.

ImpersonateUserSpecified True if the ImpersonateUser property has been specified.

OriginalConnectStringExpressionBased A read-only true/false value indicating whether the connection string for the data source is expression based.

Password A write-only property that sets the password for the data source.

Prompt The prompt to be used when prompting for credentials.

UseOriginalConnectString Set it to true to reset the original connection string.

UserName The username for the data source.

WindowsCredentials A true/false value indicating whether the Report Server passes user-provided or stored credentials as Windows credentials to the data source.

The following statements create a new data source for the AdventureWorks database and store it in the Report Server's root folder:

```
Dim name As String = "AdventureWorks"
Dim parent As String = "/"

Dim definition As New DataSourceDefinition()
definition.CredentialRetrieval =
CredentialRetrievalEnum.Integrated
definition.Extension = "SQL"
definition.ConnectString = "data source=(local);" & ➢
                           "initial catalog=AdventureWorks"
definition.Enabled = True
definition.EnabledSpecified = True
definition.ImpersonateUserSpecified = False
definition.Prompt = Nothing
definition.WindowsCredentials = False
rs.CreateDataSource(name, parent, False, definition, Nothing)
```

CreateFolder

Adds a folder to the Report Server catalog:

```
CreateFolder (Folder As String, ➢
       Parent As String, Properties As Property())
```

`Folder` is the name of the new folder, `Parent` is the full pathname of the folder to which the new folder will be added, and `Properties` is an array of `Property` objects with name/value pairs that set the properties of the folder.

The following sample code demonstrates how to create a new folder under the Report Server's root folder and assign a custom property to it:

```
Dim Prop As New [Property]()
Prop.Name = "Department"
Prop.Value = "Accounting"
Dim properties(0) As [Property]
properties(0) = Prop

Dim folderName As String = "Trial Balance"
Try
   rs.CreateFolder(folderName, "/", props)
Catch e As SoapException
   Console.WriteLine("ERROR IN CREATING FOLDER!" & ➥
                     vbcrlf & e.Detail.InnerXml)
End Try
```

CreateLinkedReport

Adds a new linked report in the specified folder:

```
CreateLinkedReport (Report As String, ➥
      Parent As String, Link As String, Properties As Property())
```

`Report` is the name of the new linked report, `Parent` is the full pathname of the folder to which the report will be added, and `Link` is the pathname of the report that will be used for the new report definition (the linked report doesn't contain its own report definition). The last parameter, `Properties`, is an array of `Property` objects that contains a set of property name/value pairs for the linked report.

Here's how you would create a linked report:

```
rs.CreateLinkedReport("New Sales Report", "/CorpReports", ➥
     "/SalesReports/Employee Sales", Nothing)
```

CreateModel

Creates a model in the specified catalog:

```
CreateModel (Model As String, Parent As String, ➥
             Definition As Byte(), Properties As Property()) ➥
             As Warning()
```

`Model` is the name of the new model, `Parent` is the pathname of the folder under which the new model will be added, and `Definition` is the report model. The last parameter is the same as for the `CreateLinkedReport` method. The `CreateModel` method returns an array of `Warning` objects that describe any warnings that may have occurred during the validation of the model definition.

CreateReport

Creates a new report in a specified folder:

```
CreateReport (Report As String, Parent As String, ➢
              Overwrite As Boolean, Definition As Byte(), ➢
              Properties As Property()) As Warning()
```

`Report` is the name of the new report, `Parent` is the path of the folder in which the report will be added, and `Overwrite` is a true/false value indicating whether an existing report with the same name will be overwritten. The `Definition` argument is an array of bytes that contains the definition of the report to be published (the report's RDL file). The last parameter is the same as for the `CreateLinkedReport` method.

The following statements create a new report based on the definition of an existing one:

```
Dim warnings As Warning() = Nothing
Dim name As String = "Test Report"
Dim stream As FileStream = File.OpenRead("MyReport.rdl")
definition = New Stream
stream.Read(definition, 0, CInt(stream.Length))
stream.Close()
warnings = rs.CreateReport(name, "/Samples", False, ➢
                           definition, Nothing)
If Not (warnings Is Nothing) Then
   Dim warning As Warning
   For Each warning In  warnings
      Console.WriteLine(warning.Message)
   Next warning
Else
   Console.WriteLine(name & "report created successfully")
End If
```

CreateReportHistorySnapshot

Generates a history snapshot of the specified report:

```
CreateReportHistorySnapshot (Report As String, ➢
         ByRef Warnings As Warning()) As String
```

`Report` is the pathname of the report and `Warnings` is an array of `Warning` objects with the warnings that occurred during report processing. The `CreateReportHistorySnaphot` method returns a string with the snapshot's datetime stamp, which is used as a unique identifier for the snapshot. Note that the warnings generated by the method are returned as a parameter passed by reference, and not as the method's return value (which is the ID of the snapshot).

CreateResource

Creates a new resource in the specified folder:

```
CreateResource (Resource As String, Parent As String, ➢
                Overwrite As Boolean, Contents As Byte(), ➢
                MimeType As String, Properties As Property())
```

`Resource` is the name of the new resource and `Parent` is the pathname of the folder in which the new resource will be added. The `Overwrite` parameter specifies whether to overwrite an existing resource with the same name. The `Contents` byte array contains the resource's data.

To add an image resource, for example, you must read the image file's contents into an array of bytes and then pass this array to the `CreateResource` method. The following statements read an image into an array of bytes:

```
Dim bytes() As Byte
bytes = File.ReadAllBytes(➢
       "c:\Windows\System32\oobe\Images\Hand1.gif")
Dim strImage As String = Convert.ToBase64String(bytes)
```

Once you have the array with the resource's contents, you can call the `CreateResource` method to publish the resource. The following statement creates a new resource (or overwrites it, if it exists) in the `Images` folder under the root folder and names it `HandPointer`:

```
CreateResource("HandPointer", "/Images", True, ➢
               bytes, "image/gif", Nothing)
```

CreateRole

Adds a new role to the Report Server catalog:

```
CreateRole (Name As String, Description As String, ➢
            Tasks As Task())
```

`Name` is the name of the new role and `Description` is the role's description. The `Tasks` array contains a `Task` object for each of the tasks to be assigned to the new role. Even though the `Task` object exposes a number of properties, only the task's ID is used in creating a new role. To find out the available tasks, call the `ListTasks` method, described later in this appendix. The following code segment creates a new role with three tasks:

```
Dim roleName As String = "Model Manager"
Dim roleDescr As String = "Manage models"
Dim newTasks(2) As Task
newTasks(0) = New Task()
newTasks(1) = New Task()
newTasks(2) = New Task()

Dim allTasks As Task() = rs.ListTasks()
Dim tsk As Task
```

```
For Each tsk In  returnedTasks
  If tsk.Name = "View Models" Then
     setTasks(0) = tsk
   Else
     If tsk.Name = "View Model Items" Then
       setTasks(1) = tsk
     Else
       If tsk.Name = "Manage Models" Then
         setTasks(2) = tsk
       End If
     End If
   End If
Next tsk
rs.CreateRole(roleName, roleDescr, setTasks)
```

CreateSchedule

Creates a new shared schedule:

```
CreateSchedule (Name As String, ➢
                ScheduleDefinition As ScheduleDefinition) As String
```

`Name` is the name of the schedule and `ScheduleDefinition` is a `ScheduleDefinition` object that defines the new schedule's properties.

The ScheduleDefinition Class

The `ScheduleDefinition` class provides the following properties that describe schedule attributes:

`EndDate` The schedule's end date and time.

`EndDateSpecified` True if the `EndDate` property was specified.

`Item` The schedule's recurrence pattern (this is a `RecurrencePattern` object, which is discussed next).

`StartDateTime` The schedule's start date and time.

The RecurrencePattern Class

The `RecurrencePattern` class is an abstract class. In its place you must use one of the following classes to specify how often the schedule will be executed: `MinuteRecurrence`, `DailyRecurrence`, `WeeklyRecurrence`, `MonthlyRecurrence`, or `MonthlyDOWRecurrence`. If you want to set up a daily schedule, use the `DailyRecurrence` class and set its `DaysInterval` property to the number of days between scheduled executions. With the `MonthlyRecurrence` class, you can specify either the days of the month on which the schedule is executed and the months of the years on which the schedule is executed. To specify the days of the month, use the `Days` property, which is a comma-separated list of days. To specify the months, use the `MonthsOfYear` property, which is a `MonthsOfYearSelector` object that exposes the names of the months as properties. Set to true the month name properties that correspond to the months for which you want to execute the report. The whole process sounds complicated, but it's much simpler once you start entering

object names followed by a period. The valid properties are displayed in the IntelliSense box, and you will quickly figure out the object hierarchy.

The MonthlyRecurrence Class

The `MonthlyRecurrence` class provides the following properties for specifying monthly schedule recurrences:

`Days` The days of the month on which a scheduled report runs.

`MonthsOfYear` The months of the year in which a scheduled report runs. This property is a `MonthsOfYearSelector` object.

The MonthsOfYearSelector Class

The `MonthsOfYearSelector` class has 12 properties (named `January`, `February`, and so on) that represent the months of the year. They are set to a true/false value that indicates whether the report should be executed in the corresponding month.

The MonthlyDOWRecurrence Class

The `MonthlyDOWRecurrence` class provides the following properties for specifying recurrences in a year's period:

`DaysOfWeek` The days of the week on which the report runs.

`MonthsOfYear` The months of the year in which the report runs.

`WhichWeek` The week of the month in which the report runs.

`WhichWeekSpecified` True if the `WhichWeek` property is specified.

The DailyRecurrence Class

The `DailyRecurrence` class describes recurrences in number of days and it provides a single property:

`DaysInterval` The intervals at which the report is executed, expressed in days.

The WeeklyRecurrence Class

The `WeeklyRecurrence` class has the following properties for describing weekly recurrences:

`DaysOfWeek` The days of the week on which the report is executed. This property is a `DaysOfWeekSelector` object.

`WeeksInterval` The intervals at which the report is executed, expressed in weeks.

`WeeksIntervalSpecified` True if the `WeeksInterval` property is specified.

The DaysOfWeekSelector Class

The `DaysOfWeekSelector` class specifies daily recurrences by day name and it has seven properties (named `Monday`, `Tuesday`, and so on) that represent the days of the week. They are set to a true/false value that indicates whether the report should be executed on the corresponding day.

The following statements set up a schedule, the DaySchedule, that's executed every working day at midnight:

```
Dim scheduleDefinition As New ScheduleDefinition()
Dim scheduleID As String = ""
scheduleDefinition.StartDateTime = ➢
             New DateTime(2007, 11, 1, 0, 0, 0)
Dim recurrence As New WeeklyRecurrence()
Dim days As New DaysOfWeekSelector()
days.Monday = True; days.Tuesday = True; days.Wednesday = True
days.Thursday = True; days.Friday = True
days.Saturday = False; days.Sunday = False
recurrence.DaysOfWeek = days
recurrence.WeeksInterval = 1
recurrence.WeeksIntervalSpecified = True
scheduleDefinition.Item = recurrence
scheduleID = rs.CreateSchedule("DaySchedule", scheduleDefinition)
```

Once you have created a ScheduleDefinition object, you can use it with the SetExecutionOptions to request that a report be executed on a schedule.

CreateSubscription

Creates a subscription for the specified report in the Report Server catalog:

```
CreateSubscription (Report As String, ➢
          ExtensionSettings As ExtensionSettings, ➢
          Description As String, EventType As String, ➢
          MatchData As String, Parameters As ParameterValue())
```

Report is the pathname of the report for which the subscription is being created, and Description is a description for the subscription. The ExtensionSettings parameters is an ExtensionSettings object that contains the settings that for the delivery extension used by the report. The EventType parameter is the type of the event that triggers the subscription, and its value can be either of these strings: TimedSubscription or SnapshotUpdated. The MatchData parameter is a string with data relating to the specified EventType parameter. This parameter matches the subscription with an event. Parameters is an array of ParameterValue objects with the report's parameters (the parameters of these reports can't be supplied when the report is executed). The CreateSubscription method returns an ID value that uniquely identifies the subscription in the Report Server's database.

There are two events you can use to create subscriptions: the TimedSubscription and SnapshotUpdated events. If the event is a TimedSubscription event, the MatchData parameter should be set to a ScheduleDefinition object, with the definition of the subscription's schedule. See the CreateSchedule entry for an example of a ScheduleDefinition object. You can also pass the schedule's definition as an XML string like the following (the "T" character in the sample is the date and time separator):

```
<ScheduleDefinition>
<StartDateTime>2007-01-10T09:00:00</StartDateTime>
```

```
<WeeklyRecurrence>
<WeeksInterval>1</WeeksInterval>
<DaysOfWeek><Monday>True</Monday>
<DaysOfWeek><Tuesday>True</Tuesday>
<DaysOfWeek><Friday>True</Friday>
</DaysOfWeek>
</WeeklyRecurrence>
</ScheduleDefinition>
```

If the schedule definition for a `TimedSubscription` is a shared schedule, you must pass its ID as the `MatchData` parameter.

DeleteItem

Deletes an item in the specified folder:

```
DeleteItem (Item As String)
```

`Item` is the pathname of the item to be deleted. To delete the report `RegionSales` in the `AWCustomers` folder, pass the string `AWCustomers/RegionSale` to the `DeleteItem` method. The `DeleteItem` method does not return a value, but if the operation fails it throws an exception. Use this method to delete reports, resources, data sources, and folders. If the item contains related items, they will be deleted as well. If you delete a report, for example, its subscriptions and schedules will be also deleted.

DeleteReportHistorySnapshot

Deletes a history snapshot for the specified report:

```
DeleteReportHistorySnapshot (Report As String, ➢
                             HistoryID As String)
```

`Report` is the pathname of the report, and `HistoryID` is the ID of the report snapshot to be deleted. The `DeleteReportHistory` method does not return a value, but it throws an exception if the operation fails.

DeleteRole

Deletes the specified role from the Report Server catalog:

```
DeleteRole (RoleName As String)
```

`RoleName` is the role's name. The `DeleteRole` method does not return a value, but it throws an exception if the operation fails.

DeleteSchedule

Deletes a shared schedule from the Report Server catalog:

```
DeleteSchedule (ScheduleID As String)
```

`ScheduleID` is the ID of the schedule to be deleted. The `DeleteSchedule` method does not return a value. To retrieve the list of all shared schedules in the Report Server catalog, call the `ListSchedules` method.

DeleteSubscription

Deletes a subscription from the Report Server catalog:

```
DeleteSubscription (SubscriptionID As String)
```

`SubscriptionID` is the ID of the subscription to be deleted. The `DeleteSubscription` method does not return a value.

DisableDataSource

Disables a data source:

```
DisableDataSource (DataSource As String)
```

`DataSource` is the pathname of the data source to be disabled. All reports that use a disabled data source will not execute. In effect, you can use this method to disable all reports that refer to a specific database.

EnableDataSource

Enables a data source that has been disabled:

```
EnableDataSource (string DataSource)
```

`DataSource` is the pathname of the data source you want to enable.

ExecuteBatch

Executes all operations associated with a batch ID in the context of a database transaction. This method accepts no arguments and returns no value, so how does the Report Server know which batch to execute? You must first set up the `BatchHeaderValue` property of the web service and then call the `ExecuteBatch` method. See the `CreateBatch` entry in this appendix for an example of creating and executing batch jobs.

FindItems

Returns the items of a specified folder that match the specified search criteria:

```
FindItems (Folder As String, ➢
           BooleanOperator As BooleanOperatorEnum, ➢
           Conditions As SearchCondition()) As CatalogItem()
```

`Folder` is the pathname of the folder to be searched. To search the entire Report Server catalog, specify the root folder (/). `BooleanOperator` is the logical operator that will be used to combine the search criteria, and its value is one of the strings AND or OR. `Conditions` is an array of

`SearchCondition` objects that contain the search criteria as pairs of property names/values. Upon completion, the method returns an array of `CatalogItem` objects that represent the matching objects.

The SearchCondition Class

The `SearchCondition` class provides the following properties:

`Condition` The type of comparison to perform between search conditions and item properties. Its value is a member of the `ConditionEnum` enumeration: `Contains` (searches on any part of the item's property value) and `Equals` (searches for an exact match).

`ConditionSpecified` Indicates whether the `Condition` property is specified.

`Name` The name of the property to be searched.

`Value` The value to search for.

The following statements search for the string `"Revenue"` in the names of all items in the Report Server's root folder:

```
Dim items As CatalogItem() = Nothing
Dim condition As New SearchCondition()
condition.Condition = ConditionEnum.Contains
condition.ConditionSpecified = True
condition.Name = "Name"
condition.Value = "Revenue"
Dim conditions(0) As SearchCondition
conditions(0) = condition
items = rs.FindItems("/", BooleanOperatorEnum.Or, conditions)
If Not (items Is Nothing) Then
   Dim itm As CatalogItem
   For Each itm In  items
      Console.WriteLine("Item " & itm.Name & ➢
                        " found at " &  itm.Path)
   Next
End If
```

FlushCache

Invalidates the cache for the specified report:

```
FlushCache (Report As String)
```

`Report` is the name of the report we want to flash from the cache.

GenerateModel

Generates a default model on top of a shared data source:

```
GenerateModel (DataSource As String, Model As String, ➢
               Parent As String, Properties As Property())
```

`DataSource` is the path to a shared data source on the server from which to generate the model, `Model` is the name of the model to create, and `Parent` is the pathname of the folder under which to add the new model. The last argument contains the new model's custom properties. The method returns an array of `Warning` objects that describe any warnings that may have occurred during the generation of the model.

GetCacheOptions

Returns the cache configuration for the specified report:

```
GetCacheOptions (Report As String, ➢
                 ByRef Item As ExpirationDefinition) As Boolean
```

`Report` is the pathname of the report, and `Item` is an `ExpirationDefinition` object that defines either the time, in minutes, or the schedule upon which the cached copy expires.
The `GetCacheOptions` method returns true if a copy of the report exists in the cache, and returns false otherwise.

GetDataDrivenSubscriptionProperties

Returns the properties of the specified data-driven subscription:

```
GetDataDrivenSubscriptionProperties (➢
        DataDrivenSubscriptionID As String, ➢
        ByRef ExtensionSettings As ExtensionSettings, ➢
        ByRef DataRetrievalPlan As DataRetrievalPlan, ➢
        ByRef Description As String, ➢
        ByRef Active As ActiveState, ➢
        ByRef Status As String, ➢
        ByRef EventType As String, ➢
        ByRef MatchData As String, _➢
        ByRef Parameters As ParameterValueOrFieldReference() ➢
) As String
```

`DataDrivenSubscriptionID` is the ID of the data-driven subscription. The second parameter is an `ExtensionSettings` object that contains the settings for the report's delivery extension. The following parameter is a `DataRetrievalPlan` object that contains the settings required to retrieve data from the delivery query for the subscription. This parameter is set by the `GetDataDrivenSubscriptionProperties` method and is passed as input to the `CreateDataDrivenSubscription` and `SetDataDrivenSubscriptionProperties` methods.

The `Description` parameter is the subscription's description, and the `Active` parameter is an `ActiveState` object that contains the subscription's active state. The `Status` parameter is a string that describes the status of the subscription, and `EventType` is the type of event that triggers the subscription. The `MatchData` parameter contains data associated with the specified type of event. This parameter is used to match the subscription with an event. The last argument, `Parameters`, is an array of `ParameterValue` objects that contains a list of parameters for the report. The `GetDataDrivenSubscriptionProperties` method returns a string with the user ID of the owner of the subscription.

GetDataSourceContents

Returns the contents of a data source:

```
GetDataSourceContents (DataSource As String) ➢
                      As DataSourceDefinition
```

`DataSource` is the pathname of the data source. The method returns a `DataSourceDefinition` object that contains the connection properties for the data source. You can retrieve the data source's connection string by calling the `ConnectString` property of the `DataSourceDefinition` object returned by this method. The properties of the `DataSourceDefinition` class are listed in the `CreateDataSource` entry, earlier in this appendix.

GetExecutionOptions

Returns the execution option and associated settings for the specified report:

```
GetExecutionOptions (Report As String, ➢
          ByRef Item As ScheduleDefinitionOrReference) ➢
          As ExecutionSettingEnum
```

`Report` is the pathname of the report, and `Item` is a `ScheduleDefinitionOrReference` object that represents a schedule definition or reference to a shared schedule. The method returns a member of the `ExecutionSettingEnum` enumeration: `Live` (the report will be executed) and `Snapshot` (the report will be rendered from a history snapshot).

GetExtensionSettings

Returns a list of settings for the specified extension:

```
GetExtensionSettings (Extension As String) As ExtensionParameter
```

The `Extension` parameter is the name of the extension (such as `Report Server Email` and `Report Server FileShare`). The method returns an array of `ExtensionParameter` objects that represent the settings of the specified extension. Use the `Name` and `Value` members of each `ExtensionParameter` object to retrieve the name and value of each extension setting.

GetItemDataSourcePrompts

Returns the data source prompts for the specified item.

```
GetItemDataSourcePrompts (Item As String) As DataSourcePrompt
```

The method accepts a single argument, which is the name of the data source for which you want to retrieve the prompts and returns an array of `DataSourcePrompt` objects. Use the `DataSourcePrompt` class's properties (listed in the `GetExectionInfo` entry at the beginning of this appendix) to retrieve information about a data source's prompt.

GetItemDataSources

Returns the data sources for an item in the catalog:

```
GetItemDataSources (Item As String) As DataSource
```

The method's parameter is the pathname of a report or model item, and its return value is an array of `DataSource` objects.

The DataSource Class

The `DataSource` class provides the following properties:

`Item` A `DataSourceDefinition` object or a `DataSourceReference` object that contains the description of the data source. The `DataSourceDefinition` class is described in the `CreateDataSource` entry of this appendix.

`Name` The name of the data source

GetItemType

Retrieves the type of an item in the Report Server catalog:

```
GetItemType (Item As String) As String
```

`Item` is the pathname of an item, and the method's return value is a member of the `ItemTypeEnum` enumeration: `Unknown` (the item is not associated with any known type), `Folder`, `Report`, `Resource`, `LinkedReport`, and `DataSource`. You can use this method in conjunction with the `ListChildren` method to find out the type of each item in a folder.

GetModelDefinition

Returns the definition of the specified model:

```
GetModelDefinition (Model As String) As Byte()
```

The method accepts as argument the pathname of a model and returns an array of Base 64-encoded bytes with the definition of the model.

GetModelItemPermissions

Returns the user permissions associated with the specified model item:

```
GetModelItemPermissions (Model As String, ➢
                         ModelItemID As String) As String()
```

`Model` is the full name of the model, and `ModelItemID` is the ID of an item in the model. Set the second parameter to `Nothing` to retrieve the permissions for the model root. The method returns an XML segment with a list of all permissions on the item.

GetModelItemPolicies

Returns the policies associate with a model item:

```
GetModelItemPolicies (Model As String, ModelItemID As String, ➢
                      ByRef InheritParent As Boolean) As Policy()
```

`Model` is the full name of the model, `ModelItemID` is the ID of the item in the model, and `InheritParent` is a true/false value indicating whether the policies are inherited from the item's parent. The method's return value is an array of `Policy` objects, each one describing the groups associated with the specified item and their roles.

The Policy Class

The `Policy` class contains the following properties:

`GroupUserName` The name of a group or user

`Roles` The roles associated with a group or user. This property is a collection of Role objects.

The Role Class

The `Roles` property of the `Policy` class is an array of `Role` objects; the `Role` class exposes the following properties:

`Description` Gets or sets the description of the role

`Name` Gets or sets the name of the role

GetPermissions

Returns the permissions associated with the specified item in the Report Server catalog:

```
GetPermissions (Item As String) As String()
```

The method accepts as argument the pathname of an item and returns an array of strings with the item's permissions. To retrieve the permissions on the `CustomerTotals` report, call the method as follows:

```
Dim permissions As String() = ➢
          rs.GetPermissions("/AWCustomers/CustomerTotals")
```

Then, you can iterate through the array returned by the method and examine each `Permission` item, which is a string like `"Execute and View"`, `"Create Subscription"`, `"Delete"`, and so on.

GetPolicies

Returns the policies associated with the specified item in the Report Server catalog:

```
GetPolicies(Item As String, ➢
            ByRef InheritParent As Boolean) As Policy()
```

`Item` is the pathname of the item whose policies you want to retrieve, and `InheritParent` is a true/false value that indicates whether the item inherits policies from its parent, set by the method. The return value is an array of `Policy` objects, each one containing the users and roles associated with the item. The `Policy` object exposes two properties: `GroupUserName` and `Roles`, as discussed in the `GetItemModelPolicies` entry in earlier in this appendix.

GetProperties

Returns the value of the specified properties of the specified item in the Report Server catalog:

```
GetProperties (Item As String, Properties As Property()) ➢
              As Property()
```

`Item` is the full pathname or the ID of the item and `Properties` is an array of `Property` objects that contains the properties whose values you want to retrieve. You must specify the property (or properties) you want to retrieve, because different items have different properties. The selected properties' values are returned in an array of `Property` objects.

GetRenderResource

Returns the resource for the specified rendering extension format:

```
GetRenderResource (Format As String, DeviceInfo As String, ➢
                   ByRef MimeType As String) As Byte
```

`Format` is the format of the rendering extension for which to retrieve the resource (this is a string like `HTML5`, `XML`, `IMAGE`, and so on), and `DeviceInfo` is an XML segment with the device-specific settings used by the rendering extension. The last argument is the resource's MIME type (this value is set by the method). The `GetRenderResource` method returns the rendering extension resource as a Base 64-encoded byte array.

GetReportDefinition

Retrieves the report definition for the report:

```
GetReportDefinition (Report As String)
```

`Report` is the pathname of the report whose definition is returned by the method as an array of Base 64-encoded bytes. To retrieve the definition of a report, call the `GetReportDefinition` method as follows:

```
Dim bytes() As Byte = ➢
       RS.GetReportDefinition("/AWReports/SalesByEmployee")
```

Once you've obtained the bytes array, you can retrieve the text of the report definition with the statement

```
System.Text.Encoding.UTF8.GetString(bytes)
```

or you can pass it to the `CreateReport` method to create a new report:

```
RS.CreateReport("SalesReport", "/New Reports", False, ➢
                bytes, Nothing)
```

The preceding statement creates an identical report, the SalesReport report, in the `New Reports` folder.

GetReportHistoryLimit

Returns the report history snapshot limit for the specified report:

```
GetReportHistoryLimit (Report As String, ➢
                       ByRef IsSystem As Boolean, ➢
                       ByRef SystemLimit As Integer) As Integer
```

`Report` is the full pathname of the report, and `IsSystem` is a true/false value set by the method, which determines whether the report snapshot policy is set equal to the system limit. It returns false if the report has its own history limit. `SystemLimit` is the system report history limit. The method returns an integer representing the history limit for the specified report.

GetReportHistoryOptions

Returns the report history snapshot option and properties for the specified report:

```
GetReportHistoryOptions (Report As String, ➢
                         ByRef KeepExecutionSnapshots As Boolean, ➢
                         ByRef Item As ScheduleDefinitionOrReference) ➢
                         As Boolean
```

`Report` is the full pathname of the report, `KeepExecutionSnapshots` is a true/false value indicating whether history snapshots are saved, and `Item` is a `ScheduleDefinitionOrReference` object set by the method and may have one of the values: `ScheduleDefinition`, `ScheduleReference`, or a `NoSchedule` object. The method returns a true/false value indicating whether Report Server keeps the history of the report.

GetReportLink

Returns the name of the report whose report definition is used for the specified linked report:

```
GetReportLink (Report As String) As String
```

The method accepts as argument the full pathname of the linked report and returns the pathname of the report that contains the definition of the linked report.

GetReportParameters

Returns and validates the parameter properties for the specified report:

```
GetReportParameters (Report As String, HistoryID As String, ➢
       ForRendering As Boolean, Values As ParameterValue(), ➢
       Credentials As DataSourceCredentials()) As ReportParameter
```

`Report` is the full pathname of the report, and `HistoryID` is the ID of the report history snapshot. Set the `ForRendering` parameter to true to retrieve parameter properties for a report history snapshot. Set the value of the `ForRendering` argument to `Nothing` to retrieve parameters for a report that is not a report history snapshot. The `Values` parameter is an array of `ParameterValue`

objects that will be validated against the report's parameters. The `Credentials` parameter is an array of `DataSourceCredentials` objects, which will be used to validate the query parameters. The method returns the report's parameters as an array of `ReportParameter` objects.

The following code segment prints the names of the parameters for a given report:

```
Dim report As String = "/AWReports/CustomerSales"
Dim historyID As String = Nothing
Dim values As ParameterValue() = Nothing
Dim credentials As DataSourceCredentials() = Nothing
Dim parameters As ReportParameter() = Nothing
parameters = rs.GetReportParameters(report, historyID, ➥
                     False, values, credentials)
If Not (parameters Is Nothing) Then
    Dim param As ReportParameter
    For Each param In parameters
        Console.WriteLine("Parameter " & param.Name & ➥
                          "Type " & param.Type.ToString)
    Next
End If
```

GetResourceContents

Retrieves the contents of a resource:

```
GetResourceContents (Resource As String, ➥
                     ByRef MimeType As String) As Byte()
```

The method accepts as argument the name of the resource and its MIME type, and returns the contents in a Base 64-encoded byte array.

GetRoleProperties

Returns role information and a collection of associated tasks:

```
GetRoleProperties (Name As String, ➥
                   ByRef Description As String) As Task()
```

This method accepts a role name and description as arguments and returns the role's properties as an array of `Task` objects. The `Task` class is presented in the `ListTasks` entry of this appendix.

GetScheduleProperties

Returns the properties of a shared schedule:

```
GetScheduleProperties (ScheduleID As String) As Schedule
```

This method accepts as argument a schedule ID and returns a Schedule object with the schedule's settings. Use the Schedule class's properties to access the properties of a schedule.

The Schedule Class

The Schedule class specifies a schedule's attributes through the following properties:

`Creator` The name of the user who created the schedule

`Definition` The schedule's definition

`Description` The schedule's description

`LastRunTime` The date and time that the schedule was last run

`LastRunTimeSpecified` True if the `LastRunTime` is specified

`Name` The schedule's name

`NextRunTime` The date and time of the schedule's next execution

`NextRunTimeSpecified` True if the `NextRunTime` property is specified

`ReferencesPresent` A true/false value indicating whether a shared schedule is referenced by reports and subscriptions

`ScheduleID` The schedule's system-generated ID

`State` The schedule's state

GetSubscriptionProperties

Returns the properties of the specified subscription:

```
GetSubscriptionProperties (SubscriptionID As String, ➢
         ByRef ExtensionSettings As ExtensionSettings, ➢
         ByRef Description As String, ➢
         ByRef Active As ActiveState, ByRef Status As String, ➢
         ByRef EventType As String, ByRef MatchData As String, ➢
   ByRef Parameters As ParameterValue()) As String
```

The first argument is the ID of the subscription. The second argument is an `ExtensionSettings` object that contains settings specific to the report's delivery extension. The `Description` parameter is the subscription's description, and the `Active` parameter is an `ActiveState` object with the active state of the subscription. The `EventType` parameter is the type of event that triggers the subscription, and `MatchData` is a string set by the method to the data associated with the specified type of event. Finally, the `Parameters` parameter is an array of `ParameterValue` objects that contains the parameter values for the subscription's report.

GetSystemPermissions

The `GetSystemPermissions` method accepts no arguments and returns the system permissions for the logged-in user in an array of strings.

GetSystemPolicies

The `GetSystemPolicies` method accepts no arguments and returns the system policies as an array of `Policy` objects. This object exposes two properties: the `GroupUserName` property (the user or group name) and the `Roles` property (a collection of `Role` objects that represent the roles of the

corresponding user or group). The system permissions are strings like `"Create Role"`, `"Update Schedules"`, and so on.

GetSystemProperties

Returns the value of one or more system properties:

```
GetSystemProperties (Properties As Property()) As Property()
```

The `Properties` array contains the names of the properties for which you want to retrieve their values. These values are returned as an array of `Property` objects, and the `Property` class exposes two properties, `Name` and `Value`, which are the name and value of a system property.

GetUserModel

Returns the semantic portion of a model for the current user:

```
GetUserModel (Model As String, Perspective As String) As Byte()
```

`Model` is the name of the model whose definition you want to retrieve, and `Perspective` is the ID of the perspective for the same model. The model contains only the parts of the model that the current user is allowed to view and is returned as an array of bytes.

InheritModelItemParentSecurity

Deletes the policies associated with a model item and causes the model item to inherit the policies from its parent:

```
InheritModelItemParentSecurity (➢
                Model As String, ModelItemID As String)
```

`Model` is the name of the model whose policies you want to reset, and `ModelItemID` is the ID of a model item. When policies are removed from an item, they are also removed from the child items that inherit them. You must specify a model item, because you can't remove policies from the entire model.

InheritParentSecurity

Deletes the policies associated with an item and causes the item to inherit the policies from its parent:

```
InheritParentSecurity (Item As String)
```

`Item` is the name of the item whose policies you want to reset.

ListChildren

Gets a list of children of a specified folder:

```
ListChildren (Item As String, Recursive As Boolean) ➢
              As CatalogItem
```

`Item` is the pathname of the parent folder and `Recursive` is a true/false value indicating whether the method will iterate recursively through all nested folders and their subfolders. The child items are returned as an array of `CatalogItem` objects. This array contains only the items that the current user is allowed to view, and not the complete list. Here's a short loop that iterates through all folders and prints their items:

```
Dim items() As CatalogItem
items = RS.ListChildren("/", True)
Dim item As CatalogItem
For Each item In items
    Console.WriteLine(item.Path & " > " & item.Description)
Next
```

The CatalogItem Class

The `CatalogItem` class provides a number of properties that describe the various items of the Report server:

`CreatedBy` The name of the user who added the item to the Report Server's catalog

`CreationDate` The date and time that the item was added to the Report Server's catalog

`CreationDateSpecified` True if the `CreationDate` property was specified

`Description` The item's description

`ExecutionDate` A read-only property with the date and time a snapshot was last created for the report

`ExecutionDateSpecified` True if the `ExecutionDate` property was specified

`Hidden` A true/false value indicating whether an item is hidden on the user interface

`HiddenSpecified` True if the `Hidden` property was specified

`ID` A read-only property with the item's ID in the Report Server's catalog

`MimeType` A read-only property with the MIME type of a resource in the Report Server's catalog

`ModifiedBy` A read-only property with the name of the user who last modified the item

`ModifiedDate` A read-only property with the date and time when the user last modified the item

`ModifiedDateSpecified` True if the `ModifiedDate` was specified

`Name` The name of an item in the Report Server's catalog

`Path` A read-only property with the pathname of an item in the Report Server's catalog

`Size` A read-only property with the size, in bytes, of an item in the Report Server's catalog

`SizeSpecified` True if the `Size` property was specified

`Type` A read-only property with the type of an item in the Report Server's catalog

`VirtualPath` A read-only property with the item's virtual path in the Report Server's catalog

ListDependentItems

Returns a list of catalog items that reference a specified catalog item:

```
ListDependentItems (Item As String) As CatalogItem()
```

The `ListDependentItems` method accepts the name of an item as argument and returns a list of dependent items as an array of `CatalogItem` objects. The dependent item types depend on the type of the specified item: if `Item` is a report, the method returns its linked reports; if it's a report model, the method returns one or more models.

ListEvents

The `ListEvents` method accepts no arguments and returns an array of Event objects, one for each of the supported events: `SnapshotUpdated`, `ReportHistorySnapshotCreated`, and `TimedSubscription`.

ListExtensions

Returns a list of extensions that are configured for the specified extension type:

```
ListExtensions (ExtensionType As ExtensionTypeEnum) As Extension
```

The `ListExtensions` method accepts a member of the `ExtensionTypeEnum` enumeration (`Delivery`, `Render`, `Data`, `All`) and returns an array of `Extension` objects, one for each of the available extensions. To find out the name of the extension, use the `Extension` object's `Name` property.

ListJobs

The `ListJobs` method accepts no arguments and returns the list of jobs running on the server as an array of `Job` objects.

The Job Class

The `Job` class exposes a number of properties for specifying jobs or retrieving the attributes of existing jobs:

`Action` The action that initiated the job (`ExecuteQuery`, `Render`, `ReportHistoryCreation`, `SnapshotCreation`)

`Description` The name of the item associated with the job

`JobID` The job ID

`Machine` The name of the computer on which a job is running

`Name` The name of the report associated with the job

`Path` The full path to the associated report

`StartDateTime` The date and time the job started

`Status` The job's status (`CancelRequested`, `New`, `Running`)

`Type` The job's type (`System`, `User`)

`User` The ID of the authenticated user who initiated the job

ListModelDrillthroughReports

Lists drillthrough reports associated with an entity in a model:

```
ListModelDrillthroughReports(➢
        Model As String, ModelItemID As String) ➢
        As ModelDrillthroughReport
```

`Model` is the name of the model, and `ModelItemID` is the ID of an item in the model for which the drillthrough reports are listed. The method returns an array of `ModelDrillthroughReport` objects. The `ModelDrillthroughReport` class exposes the `Type` and `Path` properties, which you can call to retrieve the drillthrough report's type and path.

ListModelItemChildren

Returns an array of model item child elements:

```
ListModelItemChildren (➢
        Model As String, ➢
        ModelItemID As String, Recursive As Boolean) ➢
        As ModelItem()
```

`Model` is the name of the model for which you want to retrieve child items, and `ModelItemID` is the ID of the model item whose child items will be retrieved. The last argument is a true/false value indicating whether the entire tree below the specified item should be returned. The `ListModelItemChildren` method returns an array of `ModelItem` objects, which expose the following properties:

The ModelItem Class

The `ModelItem` class provides a number of properties that describe all items of a model:

`Description` Returns the description of the item

`ID` Returns the ID of the item

`ModelItems` Returns the child items for this item

`Name` Returns the name of the item in the model

`Type` Returns the type of the item in the model

ListModelPerspectives

Lists perspectives available to the user:

```
ListModelPerspectives (Model As String) As ModelCatalogItem
```

`Model` is the name of the model for which you want to retrieve the perspectives. These perspectives are returned as an array of `ModelCatalogItem` objects. The `ModelCatalogItem` class exposes the following properties: `Description` (the perspective's description), `ID` (the perspective's ID), and `Name` (the perspective's name).

ListReportHistory

Returns a list of report history snapshots and their properties for the specified report:

```
ListReportHistory (Report As String)
```

The `ListReportHistory` method accepts as argument a report's name and returns the associated list of snapshots as an array of `ReportHistorySnapshot` objects. The `ReportHistorySnapshot` class is described next.

The ReportHistorySnapshot Class

The `ReportHistorySnapshot` class describes the attributes of a report's snapshots and it exposes the following properties:

`CreationDate` Returns or sets the date and time that the report history snapshot was created

`HistoryID` Gets or sets the ID of the report history snapshot

`Size` The size of the snapshot in bytes

ListRoles

Returns the names and descriptions of roles that the Report Server manages:

```
ListRoles (SecurityScope As SecurityScopeEnum) As Role()
```

The `ListRoles` method accepts as argument a member of the `SecurityScopeEnum` enumeration (`All`, `Catalog`, `Model`, `System`) and returns the matching roles as an array of `Role` objects. The properties of the `Role` class are listed in the `GetModelItemPolicies` entry, earlier in this appendix.

ListScheduledReports

Returns a list of reports that are associated with a shared schedule:

```
ListScheduledReports (ScheduleID As String) As CatalogItem()
```

The method accepts a schedule's ID and returns an array of `CatalogItem` objects that represent the reports associated with the specified schedule.

ListSchedules

The `ListSchedules` method does not accept any arguments and returns all shared schedules as an array of `Schedule` objects. Use the following properties of the `Schedule` class to retrieve schedule properties. The `Schedule` class's properties are discussed in the `GetScheduleProperties` entry in this appendix.

ListSubscriptions

Returns a list of subscriptions that a user has created for the specified report. The list includes both standard and data-driven subscriptions.

```
ListSubscriptions (Report As String, ➢
                   Owner As String) As Subscription()
```

`Report` is the pathname of the report, and `Owner` is the name of the user that owns the subscriptions. The subscriptions are returned as an array of `Subscription` objects.

The Subscription Class

The `Subscription` class represents the attributes of a subscription through the following properties:

`Active` The subscription's `ActiveState` object

`DeliverySettings` The settings specific to the delivery extension

`Description` A description of the format and the delivery method

`EventType` The type of event that triggers the subscription

`IsDataDriven` True for data-driven subscriptions

`LastExecuted` The date and time the subscription was last executed

`LastExecutedSpecified` True if the `LastExecuted` property is specified

`ModifiedBy` The name of the user who last modified the subscription

`ModifiedDate` The date and time that the user last modified the subscription

`Owner` The name of the subscription's owner

`Path` The pathname of the report associated with the subscription

`Report` The name of the report associated with the subscription

`Status` The status of a subscription

`SubscriptionID` The ID of the subscription

`VirtualPath` If the report associated with the subscription is located in the `My Reports` folder, contains the virtual path to the report

LISTSUBSCRIPTIONSUSINGDATASOURCE

Returns a list of subscriptions that are associated with the specified data source:

```
ListSubscriptionsUsingDataSource(➢
            DataSource As String) As Subscription()
```

`DataSource` is the pathname of a source. The subscriptions are returned as an array of `Subscription` objects that represent the user's subscriptions for the specified report. See the preceding entry for the properties of the `Subscription` class.

LISTTASKS

Returns the names and descriptions defined in the Report Server catalog:

```
ListTasks (SecurityScope As SecurityScopeEnum) As Task()
```

`SecurityScope` specifies the security scope for the tasks to be selected, and its value is a member of the `SecurityScopeEnum` enumeration: `All`, `Catalog`, `Model`, `System`. The tasks are

returned as an array of `Task` objects. Use the properties of the `Task` class to retrieve (or set) the properties of a task:

`Description` The task's description

`Name` The task's name

`TaskID` The task's ID

LogOff

The `Logoff` method accepts no argument and logs out the current user.

LogonUser

The `LogonUser` method logs on a user to the Report Server web service and must be called over SSL. If the user is logged on successfully, the method creates an authentication cookie, which is passed between the client and the web server in the request's header.

```
LogonUser (userName As String, ➢
           password As String, authority As String)
```

`userName` is the name of the user, `password` is the password of the user, and `authority` is an optional parameter that specifies the domain that will authenticate the user.

MoveItem

Moves or renames an item:

```
MoveItem (Item As String, Target As String)
```

`Item` is the pathname of the item to be moved, and `Target` is the pathname of the item's destination. The operation will fail if the user has no permission to manage items on the server.

```
PauseSchedule
```

Pauses the execution of a given schedule:

```
PauseSchedule(ScheduleID As String)
```

`ScheduleID` is the ID of the schedule to be paused. To resume schedule execution, use the `ResumeSchedule` method.

PrepareQuery

Returns a data set containing the fields retrieved by the delivery query for a data-driven subscription:

```
PrepareQuery (DataSource As DataSource, ➢
       DataSet As DataSetDefinition, ➢
       ByRef Changed As Boolean, ➢
       ByRef ParameterNames As String()) As DataSetDefinition
```

`DataSource` is the data source for the delivery query, and `DataSet` is the data set containing the delivery query and fields for a data-driven subscription. The `Changed` parameter is a true/false value indicating whether the data set that was passed to the method is different from the one returned by the method. Use this property to determine whether the input data set is different from the output data set. `ParameterNames` is an array of strings with parameter names to be used when `DataSource` references a model. The method returns a `DataSetDefinition` object, which contains the definition of the delivery query for the specified data source.

The DataSetDefinition Class

Use the following properties of the `DataSetDefinition` class to manage the delivery query:

`ConnectString` The data source's connection string

`CredentialRetrieval` Indicates the way in which the Report Server retrieves data source credentials, and its value is a member of the `CredentialRetrievalEnum` enumeration: `None`, `Prompt`, `Integrated`, or `Store`

`Enabled` True if the data source is enabled

`EnabledSpecified` True if the `Enable` property is specified

`Extension` The name of the data source extension: `SQL`, `OLEDB`, `ODBC`, or a custom extension

`ImpersonateUser` A true/false value indicating whether the Report Server should impersonate a user by using stored credentials

`ImpersonateUserSpecified` True if the `ImpersonateUser` property was specified

`OriginalConnectStringExpressionBased` A true/false value indicating whether the original connection string for the data source is expression based

`Password` A write-only property that sets the password for the specified data source

`Prompt` The prompt displayed by the Report Server when prompting for credentials

`UseOriginalConnectString` A true/false value indicating whether the data source should revert to the original connection string

`UserName` The username used by the Report Server to connect to the specified data source

`WindowsCredentials` A true/false value indicating whether the Report Server passes user-provided or stored credentials as Windows credentials to connect to the specified data source

RegenerateModel

Updates an existing model based on changes to the data source schema:

```
RegenerateModel (Model As String) As Warning()
```

`Model` is the path and name of the model to regenerate. If any warnings are generated in the process, they are returned as an array of `Warning` objects. Changing a model may break existing reports, so make sure that you don't remove/rename model entities that are used by existing reports.

RemoveAllModelItemPolicies

Deletes all policies associate with model items in the specified model:

```
RemoveAllModelItemPolicies (Model As String)
```

This method accepts the model's name as an argument and deletes all policies from the model.

ResumeSchedule

Resumes a shared schedule that has been paused:

```
ResumeSchedule (ScheduleID As String)
```

`ScheduleID` is the ID of the schedule you want to resume.

SetDataDrivenSubscriptionProperties

Sets the properties of a data-driven subscription:

```
SetDataDrivenSubscriptionProperties (_
            DataDrivenSubscriptionID As String, ➥
            ExtensionSettings As ExtensionSettings, ➥
            DataRetrievalPlan As DataRetrievalPlan, ➥
            Description As String, EventType As String, ➥
            MatchData As String, ➥
            Parameters As ParameterValueOrFieldReference())
```

`DataDrivenSubscriptionID` is the ID of the subscription, `ExtensionSettings` is an `ExtensionSettings` object that contains a list of settings specific to the delivery extension, and `DataRetrievalPlan` is a `DataRetrievalPlan` object that contains a list of settings that are required to retrieve data from the delivery query for the subscription. `Description` is a meaningful description for the schedule, `EventType` is the type of event that triggers the subscription, and `MatchData` is the data with the specified type of event.

SetDataSourceContents

Sets the contents of the specified data source:

```
SetDataSourceContents (DataSource As String, ➥
                       Definition As DataSourceDefinition)
```

`DataSource` is the pathname of the data source, and `Definition` is a `DataSourceDefinition` object that contains the definition for the data source. The members of the `DataSourceDefinition` class are discussed in the `CreateDataSource` entry of this appendix.

SetExecutionOptions

Sets execution options for the specified report:

```
SetExecutionOptions (Report As String, ➥
            ExecutionSetting As ExecutionSettingEnum, ➥
            Item As ScheduleDefinitionOrReference)
```

`Report` is the full pathname of the report, `ExecutionSetting` is a member of the `ExecutionSettingEnum` values and describes when the report should be executed, and `Item` is the schedule definition or shared schedule.

SetItemDataSources

Sets the data sources for an item in the Report Server's catalog:

```
SetItemDataSources (Item As String, DataSources As DataSource())
```

`Item` is the full name of the item for which to set the data sources, and `DataSources` is an array of `DataSource` objects.

SetModelDefinition

Sets the definition for the specified model:

```
SetModelDefinition (Model As String, ➢
                    Definition As Byte()) As Warning()
```

`Model` is the pathname of the model to which you want to apply the definition, and `Definition` is a byte array containing the XML model definition. The method returns an array of `Warning` objects that describes any warnings that may have occurred during validation of the model definition.

SetModelItemPolicies

Sets security policies on a model item:

```
SetModelItemPolicies (Model As String, ModelItemID As String, ➢
                      Policies As Policy())
```

`Model` is the full name of the model, and `ModelItemID` is the ID of a model item in the model for which to set permissions.

SetPolicies

Sets the policies associated with a specified item:

```
SetPolicies (Item As String, Policies As Policy())
```

`Item` is the pathname of the item, and `Policies` is an array of `Policy` objects that contains the groups and roles to set for the specified item.

SetProperties

Sets one or more properties of a specified item:

```
SetProperties (Item As String, Properties As Property())
```

`Item` is the pathname of the item, and `Properties` is an array of `Property` objects that defines the properties and their values.

SetReportDefinition

Sets the definition of the specified report:

```
SetReportDefinition (Report As String, ➢
                     Definition As Byte()) As Warning()
```

`Report` is the pathname of the report, and `Definition` is the report's definition.

SetReportHistoryLimit

Sets the number of snapshots of a report that the Report Server retains in the Report Server's catalog:

```
SetReportHistoryLimit (Report As String, ➢
          UseSystem As Boolean, HistoryLimit As Integer)
```

`Report` is the full pathname of the report, and `UseSystem` is a true/false expression that, when set to true, sets the report history limit to the current system report history limit. A value of false indicates that the report history limit corresponds to the value supplied in `HistoryLimit` is the number of report history snapshots to store for the report. Set `HistoryLimit` to -1 to save all snapshots.

SetReportHistoryOptions

Sets report history options that specify when a history snapshot is created for the specified report:

```
SetReportHistoryOptions (Report As String,➢
              EnableManualSnapshotCreation As Boolean, ➢
              KeepExecutionSnapshots As Boolean, ➢
              Item As ScheduleDefinitionOrReference)
```

SetReportLink

Specifies the report that is used for the report definition of an existing linked report:

```
SetReportLink (Report As String, Link As String)
```

`Report` is the pathname of the linked report, and `Link` is the pathname of the link report's definition.

SetReportParameters

Sets report parameter properties for the specified report:

```
SetReportParameters (Report As String, ➢
              Parameters As ReportParameter())
```

`Report` is the full pathname of the report, and `Parameters` is an array of `ReportParameter` objects that contains a list of the report parameters properties.

SetResourceContents

Sets the contents of an existing resource, overwriting its current contents:

```
SetResourceContents (Resource As String, ➢
       Contents As Byte(), MimeType As String)
```

`Resource` is the pathname of the resource, `Contents` is an array of bytes with the resource's contents, and `MimeType` is the MIME type of the resource. See the discussion of the `CreateResource` entry in this appendix for example of passing the resource's contents as a byte array.

SetRoleProperties

Sets role properties and associates a collection of tasks with the specified role:

```
SetRoleProperties (Name As String, Description As String, ➢
                   Tasks As Task())
```

`Name` is the name of the role for which to set properties, `Description` is the role's description, and `Tasks` is an array of `Task` objects that represents the tasks to set for the role.

SetScheduleProperties

Sets the properties of a shared schedule:

```
SetScheduleProperties (Name As String, ScheduleID As String, ➢
              ScheduleDefinition As ScheduleDefinition)
```

Name is the name of the schedule, `ScheduleID` is the `ID` of the schedule, and `ScheduleDefinition` is a `ScheduleDefinition` object that defines the schedule properties and values to set. The `ScheduleDefinition` class's properties were listed in the `CreateSchedule` entry of this appendix. To remove a property from a shared schedule, set its value to Nothing; to change the value of a property, set it to a new value.

SetSubscriptionProperties

Sets the properties of a subscription:

```
SetSubscriptionProperties (SubscriptionID As String, ➢
           ExtensionSettings As ExtensionSettings, ➢
           Description As String, EventType As String, CA
           MatchData As String, Parameters As ParameterValue())
```

`SubscriptionID` is the ID of the subscription, `ExtensionSettings` is an `ExtensionSettings` object that contains the settings specific to the delivery extension, `Description` is the

subscription's description, `EventType` is the type of event that triggers the subscription, `MatchData` is the data associated with the specified type of event, and `Parameters` is an array of `ParameterValue` objects with the report's parameters.

SetSystemPolicies

Sets the system policy that defines groups and associated roles:

```
SetSystemPolicies (Policies As Policy())
```

`Policies` is an array of `Policy` objects with the policies to set for the Report Server. The properties of the `Policy` class were listed in the `GetModelItemPolicies` entry of this appendix.

The Report Server's system properties are the following:

`SiteName` The name of the Report Server site (the name of the default installation is Microsoft Report Server).

`SystemSnapshotLimit` The maximum number of snapshots for an individual report. Set this property to 1 to save unlimited snapshots.

`SystemReportTimeout` The default report processing timeout in minutes. Individual reports may overwrite this value, which should be small and increased for specific reports that take long to execute. Set it to -1 to disable timeouts (not recommended). The default value is 5, which is already a large value, and you may wish to lower it for most reports.

`UseSessionCookies` A true/false value indicating whether Report Server should exchange session cookies with the clients (the default value is true).

`SessionTimeout` The length of time, in seconds, that a session remains active (the default value is 600).

`EnableMyReports` A true/false value indicating whether the My Reports feature is enabled.

`MyReportsRole` The name of the role to be used for creating security policies on user's `My Reports` folders (the default value is `My Reports Role`).

`EnableExecutionLogging` A true/false value indicating whether report execution is to be logged (the default value is true).

`ExecutionLogDaysKept` The number of days to keep report execution information in the log (the default value is 60). Set to 0 to disable the deletion of old entries in the log — which is not recommended.

`SnapshotCompression` Defines how snapshots are compressed, and its value is one of the strings: `SQL` — snapshots are compressed when stored in the Report Server database (default); `None` — snapshots are not compressed; or `All` — snapshots are compressed for all storage options, including the filesystem.

`EnableIntegratedSecurity` A true/false value that determines whether integrated security is supported for report data source connections (the default is false).

SetSystemProperties

Sets one or more of the system properties:

```
SetSystemProperties (Properties As Property())
```

`Properties` is an array of `Property` objects that defines the system properties and values to set for the Report Server. You can create custom system properties by specifying property names that do not exist and then use them in your code (the Report Server has no use for custom system properties).

UpdateReportExecutionSnapshot

Generates a report execution snapshot for the specified report:

```
UpdateReportExecutionSnapshot (Report As String)
```

`Report` is the pathname of the report. The method doesn't apply to `Live` reports.

ValidateExtensionSettings

Validates Reporting Services extension settings:

```
ValidateExtensionSettings (➢
        Extension As String, ➢
        ParameterValues As ParameterValueOrFieldReference()) ➢
   As ExtensionParameter()
```

`Extension` is the name of the extension as it appears in the Report Server configuration file. Valid values are `Report Server Email` and `Report Server FileShare`. `ParameterValues` is an array of `ParameterValueOrFieldReference` objects representing the settings to validate for the extension. The method returns an array of `ExtensionParameter` objects that contain validated extension settings and any required settings that were not specified.

The ExtensionParameter Class

The `ExtensionParameter` class provides the following properties that return information about an extension:

`DisplayName` The name of the extension setting

`Encrypted` A true/false value indicating whether the extension parameter value should be encrypted in the Report Server catalog

`Error` An error that describes a problem with the value of the setting

`IsPassword` Indicates whether the extension parameter value should be returned in calls to the SOAP API

`Name` The name of the device information setting

`ReadOnly` Indicates whether the setting is read-only

`Required` A true/false value indicating whether the value is required

`RequiredSpecified` True if the `Required` property is specified

`ValidValues` A set of values that can be configured for the setting

`Value` A string that represents the value of an extension parameter

Index

Note to the Reader: Throughout this index **boldfaced** page numbers indicate primary discussions of a topic. *Italicized* page numbers indicate illustrations.

A

access
  Internet, **135–136**
  .NET applications, **226–231**, *227–228*, *230*
  reports, **217–220**, **379–380**
Active Sessions counter, 339, 341
ad hoc reports, 8, **139**
  client preparation, **139–140**
  exercise, **368–369**
  Report Builder, **158–161**, *159–161*
  report models. *See* report models
  server preparation, **140–143**, *141–143*
Add a File Extension When the File Is Created option, 170
Add Counters dialog box, 117, *117*, 337, *337*
Add from Existing Site Content Types option, 406
Add Group Or User dialog box, 56–57, *56*
Add New Data Source option, 145
Add New Data Source View option, 147
Add New Item dialog box, 272
Add New Report Model option, 150
Add Reference dialog box, 246, 417
Add Subscription option, 180, 410
Add Web Parts page, 413, *413*
Add Web Reference dialog box, 228, *228*, 279
ADDLOCAL setting, 130–131
ADKAR model, 76, *77*
Advanced Settings page, 128
agendas, 64, *64*
Alert Settings dialog box, 119, *119*, 338–339
Alert view, 336
alerts, **118–120**, *119*
Analysis Services
  cubes
    building, **96–98**, *97–98*
    reports based on, **98–102**, *99–102*
  data mining, **88**
    models, **103–106**, *104–106*
    reports based on, **106–111**, *107–111*
  data sets, **323–327**, *324–326*
  installing, **89–92**, *89–92*
  OLAP, **86–88**, *87*
  projects, **92–96**, *92–96*
  requirements, 316
analytical skills for change management, **80**
anonymous access
  authentication, 257, 300, 384
  email, 135
  Report Builder directory, 143
  Report Manager, 210–211
Anonymous Access option, 257
Application log, **202–204**

Application Management page, 398–399, 402–404
archiving, **356**
authentication, **299–300**
  applications, **417–418**
  exercise, **383–384**
  proxies, 228–229
  security extensions, 257
  service accounts and passwords, **303–305**, *303–305*
  setting, **135**, **143**, *143*
  SharePoint, 403
  SSL, **301–303**
  types, **300–301**
Authentication Methods dialog box, 257
Authentication screen, 397
authoring, 7
authorization, **305–307**
  cookies, **132–133**
  folders, **307–308**, *308*
  models, **308–310**, *310*
  My Reports, **311**, *311*
  SQL injection attack prevention, **312–313**
Auto Build option, 97
automating deployment, **129**
  approaches, **129–132**
  exercise, **368**
availability in service-level agreements, 72
Available Bytes counter, 116, 337
Available Site Content Types list, 406

B

Backup dialog box, 191–192, *191–192*
backups
  encryption keys, 132
  exercise, **377–378**
  reasons, **189–190**
  types, **190–195**, *191–194*
Base64 representation, 288
baselines, performance, **356**, 388–389
Basic authentication, 384
  applications, 417–418
  limitations, 300
  proxies, 228–229
  Report Builder, 143
BeginTransaction method, 260
BIDS (Business Intelligence Development Studio), 92, 407
Binary Interchange File Format (BIFF), 10
Body element, **271**, **276**, **284–285**, **292**
branch models, 21
Browse Model task, 142, 309
Browser role, **45–46**, 306
Build the Cube Using a Data Source option, 96
bulk-logged recovery model, 190
business intelligence (BI), **85–86**
  Analysis Services. *See* Analysis Services
  data mining, **88**
  OLAP, **86–88**, *87*
Business Intelligence Development Studio (BIDS), 92, 407
business models, **17**
  company model and geographic scope, **20–21**
  exercise, **360**
  organizational structures, **21–24**, *22*
  process analysis, **17–20**
business requirements, **15–16**
  business models. *See* business models
  current, **16–17**

data-flow diagrams, **29**
dependencies, **28**
exercise, **360**
gathering, **26**
identifying, **24–25**
plans, **32–34**
priorities, **26–27**, *26*
regulatory, **30–31**
reports, **31–32**
risk tolerance, **27**
scope documents, **27–28**
training, **29–30**, *30*
business skills, change management, **80–81**

C

Cache a Temporary Copy of the Report option, 182
Cache Flushes/Sec counter, 341
Cache Hits/Sec counter, 339, 341
Cache Misses/Sec counter, 340–341
caches, 120
preloading, **375–376**
subscriptions, **181–183**, *182*
Calculated Members pane, 324
calling web service methods, **229**
Cancel method, 259
CancelBatch method, 428
CancelJob method, **428**
''Cannot create a connection to datasource'' message, 209
CatalogItem class, **449**
Central Administration tool, 397–398, *397–398*
certificates, 301
change management, **75**, *76*
change-management theory, **75–76**, *77*
exercise, **363–364**
management role, **77–78**
ongoing, **83–84**
skills, **79–81**
stages, **76**
strategies, **82–83**
chart layout, 158, 371
CheckAccess method, **250–253**, 306
Choose the Deployment Location screen, 102, 111
Choose the Installation You Want screen, 393
Choose the Table Layout screen, 102, 111
Choose the Table Style screen, 102, 111
ClearSession command, 221
ClickOnce technology, 140
client preparation for ad hoc reports, **139–140**
Close method, 260
Code property, 419
Collation Settings screen, 92
CommandText property, 259
CommandTimeout property, 259
CommandType property, 259
Committed Bytes counter, 116, 337
communication
exercise, **362**
flow processes, **19**
for user expectations, **63–65**
company model, **20–21**
company organization, **23**
company politics, **24**
compensation in service-level agreements, 72
Component Selection dialog box, 89

Components to Install screen, 89, *90*
configuration and configuration files
exercise, **368**, **388–389**
ReportingServicesService, **343–344**
RSReportDesigner, **343–346**
RSReportServer, **348–355**
RSWebApplication, **346–348**
Configuration Check screen, 89, *89*
Configure Report Server page, 401, *401*
Configure SharePoint Central Administration Web Application page, 395, *395*
Configure Sitewide Security option, 55–56, 141
Configure System-Level Role Definitions option, 49–50
Connect to a Server Farm page, 128, 394, *394*
Connection Manager dialog box, 146, *146*
connection strings
shared data sources, **318**
subscriptions, 173, 175, 182
connections
troubleshooting, **209–210**, *210*
web services, **229–230**
ConnectionString property, 260
ConnectionTimeout property, 260
ConnectionUser property, 261
Consume Reports task, 43, 142
content
managing, **197**
SharePoint types, 406
Content Manager role, **48**, 197, 306
cookies, **132–133**
costs, role-based security, **39**
counters
exercise, **387–388**
Performance Console, **337**
System Monitor, **116–120**, *117–119*
web services, **339–341**
Windows services, **341–342**
courseware, 69
Create Attributes and Hierarchies option, 96
Create Linked Reports task, 43
Create New Web Application page, 398
Create or Extend Web Application page, 398, *398*
Create Site Collection page, 399, *399*
Create the Report Server Database in SharePoint Integrated Mode option, 128, 401
CreateBatch method, **428**
CreateCommand method, 260
CreateDataDrivenSubscription method, **429**
CreateDataSource method, **429**
CreateFolder method, **231**, **430–431**
CreateLinkedReport method, **431**
CreateModel method, **431–432**
CreateParameter method, 259
CreateReport method, **232–233**, 282–283, 419, **432**
CreateReportHistorySnapshot method, **432**
CreateResource method, **433**
CreateRole method, **234–235**, **433–434**
CreateSchedule method, **434**
CreateSubscription method, **436–437**
credentials, 123. *See also* authentication
Credentials property, 221
Credentials Stored Securely on the Report Server option, 169, 175, 183

Cryptographic Services, 195–196
CSV extension, 10
  device settings, **241**
  rs utility, 220
Cube Wizard, 96–97, *97*
cubes
  building, **96–98**, *97–98*
  exercise, **365–367**, *366*
  OLAP, 86, *87*
  reports based on, **98–102**, *99–102*
customer relationships, 24
Customize Shared Documents page, 406

D

DailyRecurrence class, 435
data archiving, **356**
data definition language (DDL) statements, 88
data driven reports, 163
data-driven subscriptions
  exercise, **373–374**
  overview, **172–179**, *175–178*
data-flow diagrams, **29**, *30*
data manipulation language (DML) statements, 88
data mining, **88**
  exercise, **364–367**, *366*
  models, **103–106**, *104–106*
  reports based on, **106–111**, *107–111*
Data Mining Extensions (DMX) query language, 88
Data Mining Prediction queries, 326
Data Pane, 324
data processing extensions, 9–10, **258–259**
  deploying, **263–265**
  exercise, **381–382**
  IDataReader, **261–263**
  IDbCommand, **258–260**
  IDbConnection, **260–261**
Data Provider for mySAP Business Suite extension, 317
data sets, **315**
  Analysis Services, **323–327**, *324–326*
  defining, **321–322**
  exercise, **382**, **386–387**
  Hyperion Essbase, **332**
  Oracle Relational Database, **329–330**
  Report Model, **332–333**
  SAP Relational Database and NetWeaver BI System, **330–332**
  SQL Server Integration Services, **327–328**
  SQL Server relational database, **322–323**
  XML, **328–329**
Data Source dialog box, 323, 326, 328
Data Source Views option, 147
Data Source Wizard, 93–95, *94–95*
data sources, **315**
  defining, **315–317**, **385–386**
  exercise, **382**
  expressions, **321**
  Report Server, 198
  shared, **317–320**, *320*
Data Sources page, 198
data storage, 9
database security, 37, **310–311**
Database Setup page, 122–123, 127–129, 401
Dataset dialog box, 323, 325–326, 328
DataSet element, **290**
DataSetDefinition class, **455**
DataSets element, **274–276**, *275*

DataSource class, **442**
DataSource element, **291–292**
DataSourceDefinition class, **429–430**
DataSourcePrompt method, **421**
DataSources element, **274**, **291**
DaysOfWeekSelector class, **435–436**
DDL (data definition language) statements, 88
decentralized business units structure, 22
decision making processes, **20**
default delivery extensions, **167–168**
default security, 53
DefaultTraceSwitch setting, 205
DefaultValue element, 280
DeleteItem method, **437**
DeleteReportHistorySnapshot method, **437**
DeleteRole method, 235, **437**
DeleteSchedule method, **437–438**
DeleteSubscription method, **438**
Delivers/Sec counter, 341
delivery, 7
delivery extensions, 10–11
dependencies identification, **28**
deployment, **115**
  automating, **129–132**, **368**
  checklist, **136–137**
  data processing extensions, **263–265**, **382**
  exercise, **367–368**
  model, **154–155**, *155*
  postdeployment configuration, **132–136**, *134*
  RDL files in, **279–280**
  scaled-out, **120–125**, *121*
  security extensions, **254–258**
  SharePoint in, **125–129**
  standard, **116–120**, *117–119*
DeserializeAcl method, 251
Design the Query screen, 100, *101*, 108–110, *110*
Design the Table screen, 101, *101*, 110, *110*
Detail pane, 153
Details element, **278**, **285–286**, **294**
Detecting Fact and Dimensions screen, 97
Detecting Hierarchies screen, 97
device information settings, **240–242**, **380**
DeviceInfo argument, **240–242**
differential backups, **190**
dimensions, cubes, 86, *87*
Directory Security tab, 143, 257
DisableDataSource method, **438**
DisableLoopbackCheck option, 208
% Disk Time counter, 116, 337
Dispose method
  IDbCommand, 259
  IDbConnection, 261
DML (data manipulation language) statements, 88
DMX (Data Mining Extensions) query language, 88
DMX Designer, **326–327**, *326*
Do Not Use This Machine to Host the Website option, 128
DocMap setting
  HTML devices, 241
  rc, 224
Document Library Settings, 406, *406*
documentation for users, **73–74**
DocumentMapNode class, **420**
DpiX setting, 242
DpiY setting, 242
Dresner, Howard, 85

E
e-mail delivery
  settings, **133–134**
  subscriptions, **165–168**
Edit Page, 413
Edit Perspective dialog box, 154, *154*
Edit Role Assignment page, 55, *55*
Edit System Role Assignments page, 56
education
  exercise, **362**
  needs, **29–30**
  in user expectations, **65–69**, *67*
Elapsed Time in Minutes option, 412
elements, RDL files, **269–272**, **289–296**
Email Settings page, 166
EmbeddedImage element, **289–290**
EmbeddedImages element, 287
embedding images, **287–289**, *287*
emotion, **62–63**
empirical-rational change management strategy, **82**
Enable My Reports to Support User-Owned Folders for Publishing and Running Personalized Reports option, 199
Enable Scheduled Events and Report Delivery option, 187
Enable This Data Source option, 156
Enable Web Service and HTTP Access option, 188, 196
EnableDataSource method, **438**
Encoding setting, 241
encryption keys
  backups, 122, 132
  exercise, **378**
  managing, **195–196**
  troubleshooting, **211**
Encryption Keys page, 122
EndFind parameter, 224
EndPage setting, 242
Enter Credentials dialog box, 404, *404*
enterprise reporting, 7–8
entities, Report Model Designer, **152–153**
Error and Usage Reporting Settings screen, 397
EULA dialog box, 89
Event Viewer, 119, *119*, 335, 339
Events/Sec counter, 341
Everyone group, 54
evolutionary prototyping, 67
Excel device settings, **242**
Excel extension, 10
Execute Report Definitions task, 45, 49, 142
ExecuteBatch method, **438**
ExecuteReader method, 259–260
Execution Account page, 305, *305*
execution methods, **236–240**, *238*, *240*
Execution page, 198
ExecutionHeader class, 419
ExecutionHeaderValue property, 419
ExecutionInfo class, **420**
ExecutionTime variable, 296
Existing Components screen, 91, *91*
expectations, user. *See* user expectations
expressions
  data sources, **321**
  entity, 153
  multidimensional, **88**
''Extension'' tag, 263
ExtensionParameter class, **461**

extensions, **9–11**, **245**
data processing. *See* data processing extensions
security. *See* security extensions
types, **246**, **380–382**
external dependencies, **28**
external entities in data-flow diagrams, 29

F
FallbackPage parameter, 224
Feature Selection screen, 397
Field element, **291**
FieldCount property, 261
fields
entity, 152
RDL, 270
Fields element, **290**
file shares
extensions, 11
subscriptions, **165**
Filter Pane, 324
FindItems method, **438–439**
FindString method, **420**
FindString parameter, 224
First Session Requests/Sec counter, 340
FlushCache method, **439**
folders
creating, 231
report model, **308–309**
Report Model Designer, **153**, *153*
Report Server, **201**, *201*
security, 41, **307–308**, *308*, **384–385**
Footer element, **293–294**
Format command, **220–221**
Forms authentication, 143, **301**
From setting, 167
full backups, **190**
full recovery model, 190
functional organizational structure, 21
future, planning for, **32–34**

G
General page, 198
General Systems Theory (GST), 81
Generate Events task, 45
Generate Model page, 157
GenerateModel method, **439–440**
geographic organizational structure, 22
geographic scope, **20–21**
Get the Value from the Database option, 176–177
GetCacheOptions method, **440**
GetDataDrivenSubscriptionProperties method, **440**
GetDataSourceContents method, 220, **441**
GetDirectory method, 262
GetDocumentMap method, **420**
GetExecutionInfo method, **420–421**
GetExecutionOptions method, **441**
GetExtensionSettings method, **441**
GetFieldType method, 261
GetItemDataSourcePrompts method, **441**
GetItemDataSources method, **441–442**
GetItemType method, **442**
GetModelDefinition method, **442**
GetModelItemPermissions method, **442**
GetModelItemPolicies method, **442–443**
GetName method, 261–262
GetOrdinal method, 261
GetPermissions method, **235**, 253, **443**
GetPolicies method, **443**
GetProperties method, **444**

GetRenderResource method, **421**, **444**
GetReportDefinition method, **232**, **444**
GetReportHistoryLimit method, **445**
GetReportHistoryOptions method, **445**
GetReportLink method, **445**
GetReportParameters method, **236**, **445–446**
GetResourceContents method, 220, **446**
GetRoleProperties method, **235**, **446**
GetScheduleProperties method, **446**
GetSubscriptionProperties method, **447**
GetSystemPermissions method, **235**, **447**
GetSystemPolicies method, **447–448**
GetSystemProperties method, **448**
Getting Started pane, 159
GetUserInfo method, **248–249**, 306
GetUserModel method, **448**
GetValue method, 261
global variables in RDL files, **296**
Grant Database Access screen, 403
Graph view, 336
GST (General Systems Theory), 81

H

hardware failures, backups for, 189
hashes, 249
Header element, **276**, **293**
history, report, 43, 198
HOLAP (hybrid OLAP), 87
honesty in communication, **65**
HTML
  device settings, **241**
  injection attacks, **313**
HTML extension, 10
HTMLFragment setting, 241
HTTP errors, **207–208**
hybrid OLAP (HOLAP), 87
Hyperion Essbase data sets, **332**
Hyperion Essbase Intelligence Data Processing extension, 317

I

IAuthenticationExtension interface, **247–250**
IAuthorizationExtension interface, **250–254**
IDataParameter interface, 258
IDataParameterCollection interface, 258
IDataReader interface, 258, **261–263**
IDataReaderExtension interface, 259
IDbCommand interface, **258–260**
IDbCommandAnalysis interface, 259
IDbConnection interface, 258, **260–261**
IDbConnectionExtension interface, 258
IDbTransaction interface, 258
IDbTransactionExtension interface, 258
Identify Fact and Dimension Tables screen, 97
iisweb.vbs script, 131
Image element, **295**
Image extension, 10
images, embedding, **287–289**, *287*
Impersonate property, 261
Impersonation Information page, 94, *94*
Increment File Names as Newer Versions Are Added option, 170
Indented setting, 241
information flow processes, **18–19**
InheritModelItemParentSecurity method, **448**
InheritParentSecurity method, **448**

initialization
settings, 122–123
troubleshooting, **211**
initialization files, **129–131**
Initialization page, 122–123
injection attacks
exercise, **385**
preventing, **312–313**
installing
Analysis Services, **89–92**, *89–92*
Reporting Services, **129–131**
SharePoint, 126–127, **392–407**, *393–407*
Instance Selection dialog box, 403
INSTANCENAME setting, 130
integrated Windows authentication, **300–301**
IntegratedSecurity property, 260
integration, SharePoint
exercise, **367–368**
integrated mode, **127–129**
process, **392–407**, *393–407*
internal dependencies, **28**
international models, 20
Internet access, **135–136**
IsValidPrincipalName method, **248–250**
item-level role definitions, **50–51**, *50–51*
Item-Level Roles page, 52, *52*
item-level tasks, **42–44**
item security, **307–310**, *308*

J

Job class, **450**
jobs, 201

K

key project stakeholders, **26**
keys, encryption
backups, 122, 132
exercise, **378**
managing, **195–196**
troubleshooting, **211**
kickoff meetings, 63

L

Launch the User Provisioning Tool For Windows Vista after SP2 Installation Complete option, 397
layouts
exercise, 371
Report Builder, 158
Lewin, Kurt, 76
licensing
Reporting Services, **11**, *11–12*
SharePoint, 393, 397
lifecycles, service and product, **19–20**
LinkTarget parameter, 224
ListChildren method, 220–222, **233–234**, 418, **448–449**
ListDependentItems method, **450**
listening to users, **70–71**
ListEvents method, **450**
ListExtensions method, **450**
ListJobs method, **450**
ListModelDrillthroughReports method, **451**
ListModelItemChildren method, **451**
ListModelPerspectives method, **451**
ListRenderingExtensions method, 421
ListReportHistory method, **452**
ListRoles method, **235**, **452**
ListScheduledReports method, **452**
ListSchedules method, **452**

ListSecureMethods method, 422
ListSubscriptions method, **452–453**
ListSubscriptionsUsingDataSource method, **453**
ListTasks method, **234**, **453–454**
LoadDrillthroughTarget method, 422
LoadReport method, **422**
log files
  Report Server Execution, **206–207**
  trace logs, **205–206**
  Windows, **202–204**
Log view, 336
''Login failed for user'' message, 209
Logoff method, 422, 454
Logon.aspx page, 249
LogonUser method, **248–249**, **422**, **454**

M

Main subroutine, 221
maintenance, **185–186**
  Report Server. *See* Report Server
  troubleshooting. *See* troubleshooting
malicious destruction, backups for, 189
Manage All Subscriptions task, 43, 164
Manage Data Sources task, 43, 307
Manage Folders task, 43, 307
Manage Individual Subscriptions task, 43, 164
Manage Integration Settings option, 403
Manage Jobs task, 45
Manage Models task, 43, 142, 309
Manage Processing Options option, 412
Manage Report History task, 43
Manage Report Server Properties task, 45
Manage Report Server Security task, 45
Manage Reports task, 44, 307
Manage Resources task, 44
Manage Roles task, 45
Manage Shared Schedules task, 45
Manage Subscriptions option, 180, 409
management, 7
management model, **23**
management roles
  change management, **77–78**
  predefined, **47–49**
Management Studio
  item-level role definition, **50–51**, *50–51*
  report models, **157**, *157*
  role assignment, **56–58**, *56–58*
MarginBottom setting, 242
MarginLeft setting, 242
MarginRight setting, 242
MarginTop setting, 242
matrix layout
  exercise, 371
  Report Builder, 158
matrix organizational structure, 22
MaximumMemoryLimit setting, 357
MDX language, **88**
MDX (Multidimensional Expression) queries, **323–326**, *324–325*
Mean Time Between Failures (MTBF), 189
measurements
  performance baselines, **356**, 388–389
  in service-level agreements, 72
measures, cubes, 86, *87*
meetings, **63–65**
Memory Cache Hits/Sec counter, 340–341
Memory Cache Misses/Sec counter, 340–341
memory configuration in performance, **356–357**

MemoryLimit setting, 357
Message property, 419
messages, subscription, 181
Metadata pane
  MDX Designer, **323–325**, *324*
  Query Builder, 100
Microsoft SQL Server Analysis Services Data Processing extension, 316
Microsoft SQL Server Data Processing extension, 316
MIME (Multipurpose Internet Mail Extensions) format, 237, 287
ModelItem class, **451**
models, authorization, **308–310**, *310*, **384–385**
Modify Shared Web Part option, 414
MOLAP (multidimensional OLAP), **87–88**
monitoring. *See* performance
MonthlyDOWRecurrence class, **435**
MonthlyRecurrence class, **435**
MonthsOfYearSelector class, 435
MoveItem method, **454**
MTBF (Mean Time Between Failures), 189
Multidimensional Expression (MDX) queries, **323–326**, *324–325*
multidimensional OLAP (MOLAP), **87–88**
multiple-server environment licensing, 11
Multipurpose Internet Mail Extensions (MIME) format, 237, 287
MultiValue element, 280
My Reports feature
  enabling, **199**, *199*
  exercise, **384–385**
  securing, **311**, *311*
My Reports role, **47**, 306

N

Name attribute, 263
national models, 21
natural disasters, backups for, 189
NavigateBookmark method, **422**
NavigateDocumentMap method, **423**
.NET applications, **226–231**, *227–228*, *230*
NetWeaver BI System, **330–332**
% Network Utilization counter, 116, *337*
NetworkCredential class, 418
New Alert Settings option, 118, 338
New Cube option, 97
New Data-Driven Subscription option, 174, 182
New Data Source option, 156
New Data Source View Wizard, 95
New Database dialog box, 174
New Folder page, 201
New Mining Structure option, 103
New Role option, 50
New Role Assignment option, 54, 140
New Subscription option, 170–171
New User Role sheet, 50
Next Session Requests/Sec counter, 340–341
No, I Want to Create a New Server Farm option, 394
No Credentials option, **319**
nodes, 120, *121*
NoHeader setting, 241
normative-reeducative change management strategy, **82**
Null Delivery Provider, 182

O

Object Browser window, 230, *230*

ObjectName property, 419
ObjectType property, 419
ODBC Standard .NET Framework Data Provider extension, 317
Office SharePoint Server 2007, 126
OLAP (online analytical processing), **86**, *87*
  cubes, **86**, *87*
  exercise, **364**
  multidimensional expressions, **88**
  storage settings, **87**
OLE DB Data Processing extensions, **316–317**
OLTP (online transaction processing) queries, 85
OmitFormulas setting, 241
OmitSchema setting, 241
On a Schedule Created for This Subscription option, 178, 183
online analytical processing (OLAP), **86**, *87*
  cubes, **86**, *87*
  exercise, **364**
  multidimensional expressions, **88**
  storage settings, **87**
online transaction processing (OLTP) queries, 85
Open method, 260
Oracle Data Processing extension, 317
Oracle Relational Database data sets, **329–330**
organizational charts, 21, *22*
organizational structures, **21–24**, *22*
OutputFormat setting, 242

P

PageFooter element, **292–293**
PageHeader element, **292**
PageHeight setting, 242
PageNumber variable, 296
Pages counter, 337
Pages/Sec counter, 116
PageWidth setting, 242
ParameterLanguage command, 221
parameters
  exercise, **383**
  passing in URLs, **224–225**, *225*
  rc, 223–224
  RDL files, **280–281**
  Report Server, 198
  retrieving and setting, **236**
Parameters page, 198
Parameters setting
  HTML, 241
  IDbCommand, 259
ParameterValue Element, **290**
parties in service-level agreements, 72
partner relationships, 24
passing parameters in URLs, **224–225**, *225*, **380**
Password property, 260
passwords
  applications, 418
  changing, **303–305**, *303–305*
  hashes, 249
  IDbConnection, 260
  SharePoint, 394, 404
  subscriptions, 171
PDF extension, 10
people skills for change management, **81**
performance, **335**
  baselines, **356**, **388–389**
  configuration files
    ReportingServicesService, **343–344**

RSReportDesigner, **343–346**
RSReportServer, **348–355**
RSWebApplication, **346–348**
counters. *See* counters
data archiving and trend tracking, **356**
memory configuration, **356–357**
Performance Console, **335–342**
System Monitor, **116–120**, *117–119*
Performance Console, **335–342**
Performance Logs and Alerts, 118, 338
permissions
retrieving, **235**
role-based security, **41–48**, *42*, **56–58**
Permissions page, 56–58
perspectives in Report Model Designer, **154**
planning, **32–33**
prognostication in, **33–34**
role-based security, **35–40**
Policy class, **443**
political skills for change management, **79**
politics, **24**
port numbers for SharePoint, 396, *396*
power-coercive change management strategy, **82–83**
predefined roles, **45–49**, 306
preloading report cache
exercise, **375–376**
process, **181–183**, *182*
PrepareQuery method, **454–455**
priorities, business, **26–27**, *26*
processes
business, **17–20**
in data-flow diagrams, 29
processing options in SharePoint, **412**, *412*
% Processor Time counter, 116, 119, 337
processors, 8–9, *9*
product lifecycles, **19–20**
prognostication, **33–34**
program code, **217**
device information settings, **240–242**
execution methods, **236–240**, *238*, *240*
management methods, **231–236**
passing parameters in URLs, **224–225**, *225*
RDL files, **281–286**
report access, **217–220**
Report Server web service, **225–231**, *227–228*, *230*
rs:Command parameter, **220–224**
project stakeholders, **26**
projections, 34
Prompted Credentials option, 318–319
prototyping, **66–68**
proxies, web service, **227–229**, *227–228*
public and private keys, 195–196
Publisher role, **46–47**, 306
publishing reports with SharePoint, **407–408**, *407–408*
PublishReport method, 279, 282

Q

Query element, **291**
Query pane, 325
query parameters, 316, **383**
QueryParameter element, **291**
QueryParameters element, **291**
questions about proposed changes, **78**

R

RDL files. *See* Report Definition Language (RDL) files
Read method, 261–263

Ready to Install screen, *397*
RecordDelimiter setting, 241
RecurrencePattern class, **434–435**
reengineering business processes, **18**
refreezing stage in change management systems, 76
RegenerateModel method, **455**
regional models, 21
Registration Information screen, 89
Registry Editor, 208
regulatory requirements, **30–31**
relational OLAP (ROLAP), 87
reliability in service-level agreements, 72
RemoveAllModelItemPolicies method, **456**
RemoveSpace setting, 242
Render method, 220, **237–239**, *238*, *240*, **423–424**
rendering extensions, 10
RenderStream method, **425**
renegotiation in service-level agreements, 72
Report Builder, **158**
  exercise, **371**
  report creation and design, **158–161**, *159–161*
Report Builder role, **46**, 306
report caches, 120
  preloading, **375–376**
  subscriptions, **181–183**, *182*
Report Definition Language (RDL) files, **267**, 406
  contents, **267–268**, *268*
  creating, **272–278**, *273*, *275*
  in deployment, **279–280**
  elements, **289–296**
  embedding images, **287–289**, *287*
  global variables, **296**
  parameters, **280–281**
  program code, **281–286**
  structure, **269–272**
report definitions
  exercise, **382**
  retrieving, **232**
Report Designer, data processing extension deployment to, **264–265**, **386–387**
Report element, **269**, **289**
Report History Default setting, 404
Report Manager
  report models, **156–157**, *156–157*
  role assignments, **54–56**, *54–55*
  system-level role definitions, **49–50**, *49*
  troubleshooting, **210–211**
Report Manager Virtual Directory page, 122
Report Manager Web Service Identity page, 122
Report Model Data Sources extension, 317
Report Model Designer, **144–145**
  entities, **152–153**
  folders, **153**, *153*
  model deployment, **154–155**, *155*
  perspectives, **154**
  projects, **145–152**, *145–152*
report models, **143–144**, *144*
  data sets, **332–333**
  exercise, **369–371**
  Management Studio, **157**, *157*
  Report Manager, **156–157**, *156–157*
  Report Model Designer. *See* Report Model Designer
  security, 41
report processing extensions, 10

Report Processing Log setting, 404
Report Processing Timeout setting, 404
Report Requests counter, 340, 342
Report Server, **186**
  backups
    reasons, **189–190**
    types, **190–195**, *191–194*
  components, 8–9, *9*
  encryption keys, **195–196**
  folders, **201**, *201*
  My Reports, **199**
  processes, **201–202**
  resource management, **199–200**
  service administration, **186–189**, *188*
  settings, **198–199**, *198*
''Report Server cannot open a connection to the Report Server database'' message, 209
Report Server Execution logs, **206–207**
Report Server Project Wizard, 98–102, *100–102*, 107
Report Server Virtual Directory page, 122, 127
Report Server web service
  exercise, **380**
  .NET applications access, **226–231**, *227–228*, *230*
Report view in Performance Console, 336
ReportExecution2005 Web Service, **417–426**
ReportFolder variable, 296
ReportHistorySnapshot class, **452**
reporting infrastructure, **3**
  benefits, **6–8**
  exercise, **359–360**
  extensions, **9–11**
  licensing, **11**, *11–12*
  need, **5–6**
  overview, **3–4**, *4*
Reporting Services Configuration tool
  e-mail delivery, 133, *134*, 166
  scaled-out deployment, 122
  service accounts, 303, 305
  SharePoint installation, 126–127
ReportingServicesService file, **343–344**, 388
ReportItem element, **269–271**
ReportItems element, **383**
ReportName variable, 296
ReportParameter class, **426–427**
ReportParameter element, **290**
ReportParameters element, 280, **290**
Reports Executed/Sec counter, 340, 342
Reports Properties dialog box, 257
ReportServerTempdb database, 190
ReportService Methods, **428–462**
ReportService2005 Web Service, 417, **427–462**
Requests/Sec counter, 340
Require Secure Socket Layer Connections option, 301
requirements, business. *See* business requirements
ResetExecution method, 425
resources
  managing, 44, **199–200**
  security, 41
Restore Database dialog box, 193, *193–194*
RESTORE WITH RECOVERY option, 193
restoring database
  exercise, **377–378**
  process, 193, *193–194*
Result pane, 325
ResumeSchedule method, **456**
Review New Dimensions screen, 97

risk factors, **40**
risk tolerance, **27**
ROLAP (relational OLAP), *87*
role-based security, **35**
  assignments, **52–58**, *53–57*
  benefits, **39–40**
  costs, **39**
  customizing roles, **49–52**, *50–52*
  definitions, **44–45**
  exercise, **360–362**
  permissions, **41–48**, *42*, **56–58**
  planning, **35–40**
  predefined roles, **45–49**
  requirements
    analysis, **37–38**
    conflicts, **38–39**
    gathering, **36**
  risk factors, **40**
  scope, **37**
  securable items, **41**
  tasks and permissions, **41–44**, *42*
Role class, **443**
roles
  creating, **234–235**
  deleting, 235
  entity, 152
  predefined, 306
  properties, **235–236**
  retrieving, **235**
rs utility, 418
  exercise, **380**
  parameters, **220–224**
RSACCOUNT setting, 130
RSAUTOSTART setting, 130
RSCONFIGURATION setting, 130
.rsds files, 406
rskeymgmt.exe tool, 196, 378
RSPASSWORD setting, 130
RSReportDesigner.config file, **343–346**, 388
RSReportServer.config file
  backing up, 124
  data processing extension, 263, 382
  e-mail, 132–135
  encryption, 195–196, 378
  settings, **348–355**, 388
  SharePoint, 396
  SSL, 303
  subscriptions, 166
  Windows service memory, 357
.rss extension, 418
RSSQLLOCAL setting, 130
rssrvpolicy.config file, 256, 264
RSTrace settings, **205–206**
RSUPGRADEDATABASEACCOUNT setting, 130
RSUPGRADEPASSWORD setting, 130
RSWebApplication.config file
  in deployment, 124–125
  settings, **346–348**, 388
  subscriptions, 167–168
Run As command, 209
Running Processes screen, 397

S

SAP NetWeaver Business Intelligence, 317–318, **330–332**
Sarbanes-Oxley Act, 32, 361
satisfaction surveys, 69
scaled-out deployment
  exercise, 367
  process, **120–125**, *121*
Schedule class, **447**

Scheduled Events and Report Delivery option, 187
ScheduleDefinition class, **434**
scheduling and delivery processor, 9
schema scope in security, *37*
scope criteria in security, **37**
scope documents, **27–28**
SearchCondition class, **439**
Section parameter, 224
Secure Individual Model Items Independently for This Model option, 309
Secure Sockets Layer (SSL)
configuring for, **301–303**
exercise, **384**
SecureConnection Levels, **302**
security, 7, **299**
authentication. *See* authentication
authorization. *See* authorization
exercise, **360–362**, **384–385**
role-based. *See* role-based security
website, 208
security extensions, 9, **246–247**
deploying, **254–258**
exercise, **381**
IAuthenticationExtension, **247–250**
IAuthorizationExtension, **250–254**
Security page, 54, 140–141, 198, 210
Select a Data Source screen, 95, 147, *148*
Select a Data Source View screen, 150, *150*
Select a Source of Data for Your Report option, 159
Select an Item page, 414, *414*
Select Backup Destination dialog box, 191, *191*
Select Build Method screen, 96, *976*
Select Data Source View screen, 97, 103
Select How To Define The Connection page, 145, *146*
Select Measures screen, 97
Select Mining Model dialog box, 108, *109*
Select Report Model Generation Rules screen, 150, *150*
Select Report Type screen, 101, 110
Select Tables and Views screen, 148, *148*
Select the Backup Sets to Restore setting, 193
Select the Data Mining Technique setting, 103
Select the Data Source screen, *99*, *100*, 107, *108*
Select the Definition Method screen, 103
Semantic Model Definition Language (SMDL), 144
SendEmailToUserAlias option, 133
SendUsing setting, 166
Server Status page, 405
Server Type screen, 393, *393*
ServerBtnLogon method, 250
ServerInfoHeader class, 419–420
ServerInfoHeaderValue property, **419–420**
servers
ad hoc reports, **140–143**, *141–143*
Report Server. *See* Report Server
in security criteria, *37*
subscriptions, **165–168**
Service account screen, 91, *91*
service accounts
Analysis services, 91, *91*
changing, **303–305**, *303–305*
Service Credentials option, 129
service-level agreements (SLAs)
elements, **71**

exercise, **362–363**
preparing, **72**
services
Analysis Services. *See* Analysis Services
lifecycles, **19–20**
in security criteria, 37
in service-level agreements, 72
web. *See* web services
Windows, 186–187
counters, **341–342**
exercise, **376–377**
memory configuration, **357**
session state information, 120
Set Security for Individual Items task, 44
Set Security on Items task, 307
Set Server Defaults option, 404
SetConfiguration method, 247, 254
SetDataDrivenSubscriptionProperties method, **456**
SetDataSourceContents method, **456**
SetExecutionCredentials method, **425**
SetExecutionOptions method, **456–457**
SetExecutionParameters method, **236–237**, **425–426**
SetItemDataSources method, **457**
SetModelDefinition method, **457**
SetModelItemPolicies method, **457**
SetPolicies method, **457**
SetProperties method, **457**
SetReportDefinition method, **458**
SetReportHistoryLimit method, **458**
SetReportHistoryOptions method, **458**
SetReportLink method, **458**
SetReportParameters method, 236, **458–459**
SetResourceContents method, **459**
SetRoleProperties method, **235–236**, **459**
sets, data. *See* data sets
SetScheduleProperties method, **459**
SetSubscriptionProperties method, **459–460**
SetSystemPolicies method, **460**
SetSystemProperties method, **460–461**
Severity property, 419
Share This Folder option, 165
Shared Data Source dialog box, 168
shared data sources
creating, **317–320**, *320*
security, 41
subscriptions, 168
SharePoint
installing, **125–127**
integration
exercise, **367–368**
integrated mode, **127–129**
process, **392–407**, *393–407*
processing options, **412**, *412*
publishing reports, **407–408**, *407–408*
requirements and features, **391–392**
subscriptions, **179–180**, **408–412**, *409–411*
web parts, **412–415**, *413–415*
SharePoint Configuration Wizard, 393–394, *393*
SharePoint Products and Technologies Configuration Wizard, 128
SharePointRS.msi file, 402
side-by-side Reporting Service/SharePoint installation, **126–127**
Simple Object Access Protocol (SOAP) over HTTP, 227
simple recovery model, 190
SimplePageHeaders setting, 242

single-server environment licensing, 11, *11*
single sign-on (SSO) application, **132–133**
Site Settings page, 201
skills for change management, **79–81**
SLAs (service-level agreements)
  elements, **71**
  exercise, **362–363**
  preparing, **72**
SMDL (Semantic Model Definition Language), 144
.smdl files, 406
SMTP
  port setting, 134
  servers, **135**
SMTPAuthenticate setting, 166
Snapshot command, 221
snapshots, 120
SnapshotUpdated event, 436
SOAP (Simple Object Access Protocol) over HTTP, 227
soft skills for change management, 79
Solution Property Pages dialog box, 168
Sort method, **427**
sources, data. *See* data sources
Specify a Static Value option, 176
Specify an Execution Account option, 305
Specify Backup dialog box, 193
Specify Columns' Content and Data Type screen, 106
Specify Configuration Database Settings page, 128, 394, *394*
Specify for This Subscription Only option, 174, 182
Specify Table Types screen, 103, *104*
Specify the Training Data screen, 104, *104–105*
SQL injection attacks, preventing, **312–313**
SQL Server 2005 Service Pack 2, 396
SQL Server Agent service, **189**
SQL Server Analysis Services. *See* Analysis Services
SQL Server Connection dialog box, 122–123, 128, 401, *401*
SQL Server Integration Services (SSIS), **327–328**
SQL Server relational database data sets, **322–323**
SQL Server Reporting Services Report Viewer option, 413
SQL Server Surface Area Configuration tool, 187
  connections, 209
  Internet access, 136
  scaled-out deployment, 123
  security, 137
  for services, 187–189, *188*
SSIS (SQL Server Integration Services), **327–328**
SSL (Secure Sockets Layer)
  configuring for, **301–303**
  exercise, **384**
SSO (single sign-on) application, **132–133**
stakeholders, identifying, **26**
standard deployment
  exercise, 367
  planning, **116–120**, *117–119*
standard subscriptions, 163
  creating, **169–172**, *169–172*
  exercise, **372**
StartFind parameter, 224
StartPage setting, 242
State property, 260

status messages for subscriptions, 181
Stored Credentials option, **319**, 409
StoredProcedure command type, 322
storyboards, **66–67**, *67*
structure models, **21–22**, *22*
Style element, 268, 270, **278**, **295–296**
Subscription class, **453**
Subscription Properties page, 411, *411*
subscriptions, **163**
  creating, **168–169**
  data-driven, **172–179**, *175–178*
  e-mail delivery, **165–168**
  exercise, **371–376**
  file shares, **165**
  managing, **180–181**
  overview, **164–165**
  report caches, **181–183**, *182*
  servers, **165–168**
  SharePoint with, **179–180**, **408–412**, *409–411*
  standard, **169–172**, *169–172*
  troubleshooting, **211**
Subscriptions tab, 174, 182
subsidiary models, 21
Suggest Related Columns screen, 105, *105*
surveys, **69–70**
symmetric keys, 195–196
System Administrator role, **48–49**, 187, 306
system jobs, 201
system-level roles, **49–50**, *49*, 52, *52*
system-level tasks, 42, **45**
System Monitor counters, **116–120**, *117–119*
System Role Assignments page, 55–56, *55*, 141, *141*
system skills for change management, **81**
System User role, **47**, 306

T

Table element, **293**
table layout, 158, 371
TableCell element, **295**
TableCells element, **294**
TableColumn element, **293**
TableColumns element, **293**
TableDirect command type, 322
TableGroup element, **294**
TableGroups element, **294**
TableRow element, **294**
TableRows element, **294**
Task Manager, 335
tasks
  retrieving, **234**
  role-based security, **41–44**, *42*
Text command type, 322
Textbox element, 272, **277**, **295**
throwaway prototyping, 67
TimedSubscription event, 436
timeliness in service-level agreements, 72
timestamp.log files, 205
To field, 133, 166
ToBase64String method, 288
ToggleItem method, 427
Toolbar setting
  HTML, 241
  rc, 223
Total App Domain Recycles counter, 342
Total Cache Flushes counter, 342
Total Cache Hits counter, 340, 342
Total Cache Misses counter, 340, 342
Total Deliveries counter, 342
Total Events counter, 342
Total Memory Cache Hits counter, 340, 342

Total Memory Cache Misses counter, 340, 342
Total Processing Failures counter, 340, 342
Total Rejected Threads counter, 340, 342
Total Reports Executed counter, 341–342
Total Requests counter, 341–342
TotalPages variable, 296
trace logs, **205–206**
training
  exercise, **362**
  needs, **29–30**
  in user expectations, **65–69**, *67*
training classes, **68–69**
transaction log backups, **190**
Transaction property, 259
Tree View, 152–154, *152*
trend analysis, 34, **356**
troubleshooting, **202**
  configuration, **211–212**
  connections, **209–210**, *210*
  exercise, **379**
  HTTP errors, **207–208**
  initialization and encryption, **211**
  log files for, **202–207**
  Report Manager, **210–211**
  report processing, **209**
  reports, **208–209**
  subscriptions, **211**
Trusted Authentication, 403
Type attribute, 263

U

UDDI (Universal Description, Discovery, and Integration) servers, 227
unattended report processing, **305**, *305*
unfreezing stage in change management systems, 76
Universal Description, Discovery, and Integration (UDDI) servers, 227
Update Model Statistics Before Generating option, 151
Update Statistics screen, 151
UpdateReportExecutionSnapshot method, **461**
URLRoot setting, 396
URLs
  passing parameters in, **224–225**, *225*
  report access with, **218–220**, **379–380**
  Report Builder, 158
Use as Windows Credentials option, 409
Use as Windows Credentials When Connecting to the Data Source option, 169, 175, 183, 318
Use Cached Data option, 412
Use Existing Application Pool option, 398
Use the Built-in System Account option, 91
Use Windows Authentication option, 146
user expectations, **61–62**, *61*
  communication for, **63–65**
  documentation for, **73–74**
  education in, **65–69**, *67*
  listening to users, **70–71**
  service-level agreements, **71–72**
  understanding users, **62–63**
user jobs, 201
user requirements, **25**
UserName property, 260
usernames
  applications, 418
  SharePoint, 394, 404

subscriptions, 171
validity, 249
Using Both TCP/IP and Named Pipes option, 210
Using Impersonation with Stored Credentials option, 319

V

VAK (visual-audio-kinesthetic) learning style, **66**
ValidateExtensionSettings method, **461**
validation keys in scaled-out deployment, 124
validity of user names, 249
ValidValues element, 280
variables in RDL files, **296**
vendor relationships, 24
VerifyPassword method, 248–250
VerifyUser method, 248
View Data Sources task, 44
View Folders task, 44, 307
View Models task, 44, 50, 142, 309
View Report Server Properties task, 45, 49
View Reports task, 44, 50
View Resources task, 44
View Shared Schedules task, 45
viewing reports, **218–220**
virtual directories, 131
Visible attribute, 263
visual-audio-kinesthetic (VAK) learning style, **66**
Visual Basic for RDL files, **282–284**
volume in service-level agreements, 72

W

Warning class, 418–419
web-based reporting, 8
web farm licensing, 11
web parts in SharePoint, **412–415**, *413–415*
web services, 187
counters, **339–341**
exercise, **376–377**, **380**
memory configuration, **356–357**
proxies, **227–229**, *227–228*
website configuration, **131**
weekly meetings, 64
WeeklyRecurrence class, **435**
Windows authentication, 143
applications, 417
integrated, **300–301**
SharePoint, 403
Windows Credentials option, 129
Windows Integrated Security option, 319
Windows Load Balancing software, 121
Windows log files, **202–204**
Windows services, 186–187
counters, **341–342**
exercise, **376–377**
memory configuration, **357**
Windows SharePoint Services (WSS). *See* SharePoint
WriteAttributeString method, 281
WriteElementString method, 281–282
WriteEndElement method, 281
WriteStartElement method, 281
WSS (Windows SharePoint Services). *See* SharePoint

X

XML
    data sets, **328–329**
    device settings, **241**
    extensions, 10, 317
XmlTextWriter class, **281–282**, **383**
XSLT setting, 241

Z

Zoom setting
    HTML, 241
    rc, 224