Pro SQL Server 2005 Reporting Services

Rodney Landrum
and Walter J. Voytek II

Apress®

Pro SQL Server 2005 Reporting Services

Copyright © 2006 by Rodney Landrum and Walter J. Voytek II

ISBN (pbk): 1-59059-498-3

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Tony Davis
Technical Reviewer: Trish Middleton, Chris Rausch, Thomas Rizzo
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Project Manager: Sofia Marchant
Copy Edit Manager: Nicole LeClerc
Copy Editor: Kim Wimpsett, Julie McNamee
Assistant Production Director: Kari Brooks-Copony
Production Editor: Laura Cheu
Compositor and Artist: Kinetic Publishing Services, LLC
Proofreader: April Eddy, Linda Seifert
Indexer: Broccoli Information Management
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code section.

To all the victims of hurricanes Dennis, Katrina, and Rita,
which ravaged the Gulf Coast during the 2005 hurricane season.
Rodney Landrum andWalter J. Voytek II

Contents at a Glance

About the Authors xiii
About the Technical Reviewers. xv
Acknowledgments xvii
Introduction. xix

CHAPTER 1 Introducing the Reporting Services Architecture 1
CHAPTER 2 Report Authoring: Designing Efficient Queries 17
CHAPTER 3 Using Report Designer 35
CHAPTER 4 Building Reports. 81
CHAPTER 5 Using Custom .NET Code with Reports 123
CHAPTER 6 Rendering Reports from .NET Applications. 147
CHAPTER 7 Deploying Reports 175
CHAPTER 8 Managing Reports 197
CHAPTER 9 Securing Reports 249
CHAPTER 10 Delivering Business Intelligence with SSRS. 285
CHAPTER 11 Performing Ad Hoc Reporting Using Report Builder 321

INDEX 363

Contents

About the Authors xiii
About the Technical Reviewers xv
Acknowledgments xvii
Introduction xix

■CHAPTER 1 **Introducing the Reporting Services Architecture** 1

Understanding the Benefits of SSRS 2
  SQL Server 2005 Reporting Services Enhancements 4
  SSRS and Business Intelligence 6
Exploring the SSRS Architecture 7
  SSRS Databases 8
  The SSRS Report Server 8
  Client Applications 10
Installing and Configuring 13
Deploying SSRS Securely 14
Summary 15

■CHAPTER 2 **Report Authoring: Designing Efficient Queries** 17

Introducing the Sample Relational Database 18
  Introducing the Schema Design 18
  Knowing Your Data: A Quick Trick with a Small Procedure 19
Introducing Query Design Basics 21
  Creating a Simple Query Graphically 21
  Creating an Advanced Query 24
Using a Parameterized Stored Procedure 30
  Using Case and ISNULL to Evaluate the Parameters 32
  Testing the Procedure 33
Summary 34

CHAPTER 3 Using Report Designer 35

Exploring the Elements of BIDS. 36
- Setting Up a Basic IDE 38
- Understanding Report Definition Language (RDL) 39
- Adding a Report 40
- Setting Up Data Sources and Datasets 41
- Setting Parameters 46
- Setting Up Filters 48
- Expressions. 50

Laying Out a Report 51
- Setting Up Pagination 52
- Using Report Objects. 53
- Implementing Two Simple Report Design Tips 77

Summary 79

CHAPTER 4 Building Reports 81

Creating a Report with the Report Wizard. 82
Building Reports from Scratch 85
- Formatting the Output. 87
- Adding Subtotals 88
- Adding Interactivity 89

Setting Report Parameters with Stored Procedures 105
- Applying a Filter 111
- Adding a Chart 112

Adding the Final Touches 114
Working with Multivalued Parameters. 116
Summary 121

CHAPTER 5 Using Custom .NET Code with Reports 123

Using Embedded Code in Your Report 124
- Using the ExceedMaxVisits Function 125
- Using the ExceedMaxVisits Function in a Report 128
- Accessing .NET Assemblies from Embedded Code 131

Using Custom Assemblies with Your Report 132
- Adding a Class Library Project to Your Reporting Solution 133
- Deploying a Custom Assembly 136
- Adding an Assembly Reference to a Report 140
- Debugging Custom Assemblies. 143
- Troubleshooting Your Project 145

Summary 146

CHAPTER 6 **Rendering Reports from .NET Applications** 147

Implementing URL Access 149
URL Report Access Path Format 150
URL Parameters and Prefixes 150
Report Parameters. 151
HTML Viewer Commands 151
Report Server Command Parameters. 151
Credential Parameters 152
Example URLs. 152
Integrating SSRS 2005 with .NET Applications 153
Building the Report Viewer Using a WebBrowser Control 153
Building the Report Viewer Using a Report Viewer Control. 155
Summary 173

CHAPTER 7 **Deploying Reports** 175

Using Report Manager 176
Using SQL Server Management Studio (SSMS) 179
Using BIDS and Visual Studio 2005 181
Configuring Report Deployment Options 181
Using the Report Server Web Service 184
Accessing the Web Service 185
Laying Out the Form 186
Coding the Form. 187
Running the Application 193
Summary 195

CHAPTER 8 **Managing Reports** 197

Exploring Management Roles in SSRS Deployment 197
Managing Content. 198
Setting Up Shared Schedules 198
Setting Up a Data Source for the Report 205
Creating Snapshots for the Report History 206
Executing Reports and Performing Caching 209
Managing Subscriptions. 211
Using SSMS and SSRS 222
Performing Execution Auditing and Performance Analysis. 225
Configuring SSRS Logging 226
Monitoring Performance. 231

Controlling SSRS Programmatically 234
Controlling SSRS with SOAP 234
Controlling SSRS with WMI 245
Summary 246

CHAPTER 9 **Securing Reports** 249

Encrypting Data 250
Introducing Encryption 250
Securing Network Traffic Using SSL 251
Setting Up Authentication and User Access to Data 261
Introducing SSRS Roles 262
Testing SSRS Role Assignments 267
Filtering Report Content with User!UserID 273
Setting Data Source Security 275
Setting SQL Server Permissions 276
Auditing Reports 277
Introducing SSRS Auditing 277
Introducing Log File Auditing 278
Exploring Deployment Models 278
Implementing SSRS with Terminal Services 279
Implementing for Internal Access 282
Summary 283

CHAPTER 10 **Delivering Business Intelligence with SSRS** 285

Extending Microsoft CRM with SSRS 286
Using the Report Pack for Microsoft CRM 1.2 287
Creating Custom SSRS Reports for Microsoft CRM 291
Creating the Sales Projections Chart Report 292
Creating the Lead Conversion Report 293
Managing Projects with SSRS 294
Building SSRS Reports for SQL Analysis Services 296
Using Analysis Service Cube with SSRS 301
Setting Up the Analysis Services Data Source 301
Working with the Graphical MDX Query Builder 303
Incorporating SSRS with SharePoint Portal Server 308
Creating a Web Part Page 308
Adding Web Parts 310
Summary 319

CHAPTER 11 Performing Ad Hoc Reporting Using Report Builder 321
Getting User Feedback 322
Introducing the Report Model 324
Adding a Report Model to BIDS 324
Adding a Data Source 325
Creating a Data Source View 328
Creating a Report Model. 334
Using Report Builder 341
Creating a Table Report. 344
Adding a Matrix Report 351
Adding a Chart Report. 357
Summary 362

INDEX 363

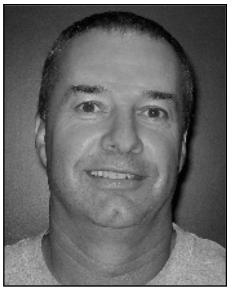
About the Authors

RODNEY LANDRUM is an MCSE working as a systems engineer, DBA, and data analyst for a software development company in Pensacola, Florida, that specializes in applications for the health-care industry. He writes software reviews and feature articles for numerous magazines, including *Windows & .NET Magazine*, *SQL Server Magazine*, *Connected Home*, *T-SQL Solutions*, *Microsoft Certified Professional Magazine*, and *Electronic House*.

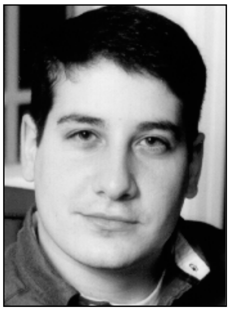
WALTER J. VOYTEK II (JIM) is the CEO and president of HealthWare Corporation, a Microsoft Certified Partner, which specializes in information technology solutions for the health-care industry. He has worked in information technology for more than 30 years and in health-care IT for nearly 20 years. He has spoken publicly at several national conventions and also speaks for HealthWare in a variety of settings each year. As the founder and chief software architect for HealthWare, Jim has been instrumental in the design and development of HealthWare's award-winning solutions based on Microsoft technologies.

About the Technical Reviewers

TRISH MIDDLETON is a professional software developer with more than 15 years of experience. She is currently focused on C#, HTML, XSLT, XML, and SQL Server while developing a Web application for Travis Software. She obtained her MCSD and MCDBA last year while taking a sabbatical from the corporate world. Outside of computers, Trish likes to run, rollerblade, do yard work, and do home improvements. When she isn't doing those things, she's playing with her three dogs—Toppsin, Backspin, and Juliet.



CHRIS RAUSCH has been a software engineer for the *Sheridan Press* for more than eight years and has more than eleven years' programming experience. Chris has developed for the Unix/Solaris and Windows platforms, creating Web applications and GUI-driven Windows applications using multiple languages, data stores, and protocols. Most recently, he has accepted the role of project manager for several digital technology projects.



THOMAS RIZZO is a director in the SQL Server group and an 11-year veteran at Microsoft. Beyond his work in SQL Server, Thomas has worked in various server groups at Microsoft, including Exchange, SharePoint, and BizTalk Server. He is also the author of books about programming using Microsoft's collaborative technologies, and he recently coauthored *Pro SQL Server 2005* (Apress, 2005). You can reach Thomas at `thomriz@microsoft.com`.

Acknowledgments

I would first like to thank my father, who, when I was 13, wanted me to be a systems analysts like him. I didn't know what that was exactly, but I knew that there were computers involved. And computers meant games. When he bought my first Atari, with a 300-baud modem and a Microsoft Basic Programming cartridge, I knew his intentions were good. Fifteen years later, when I had become the modern equivalent of a systems analyst, he didn't stop me from getting an Atari tattoo, though he laughed the whole way through. So, I want to thank my dad for his investment in my future and for his continuing encouragement.

I cannot thank enough my mother, Faye, for her support while I was writing this book and during many other times. Knowing that I had three small children, a full-time profession, and a book to write on a tight deadline constantly made her ask, "How do you do it?" I can say now, "Not without you, Mom!"

I would also like to thank Karla, who has been a constant source of inspiration and encouragement, keeping me heads-down at the computer when I would rather have been shooting pool—and for understanding while I worked on Chapter 2 in the train station in Madrid while on vacation.

Jim, thank you for being the coauthor of this book. I enjoyed all the book meetings. You did an excellent job.

I also would like to thank Eric Doverspike, who turned me on to the creative uses of `ISNULL`. Eric is a dedicated .NET programmer with the sort of sound work ethic that I envy. I wish I could get to work as early as he does. But 5:30 a.m. is sleepy time.

Finally, I would like to thank everyone at Apress who played important roles in shaping the book. It's immensely better, thanks to all your careful attention. Special thanks go to Tony Davis, lead editor, and Tom Rizzo, technical reviewer, whose thoughtful comments not only helped the book but made me a better writer. Thanks to Laura Cheu, who, under a pressured deadline, kept everything moving swiftly forward and brought it all together. Hats off to the copy editors, Kim Wimpsett and Julie McNamee, who leapfrogged over many hurdles to insure—I mean, *ensure*—the text made sense.

You all did a great job!

Rodney Landrum

I would like to thank Rodney, my coauthor, because without him I would not have been involved with this book at all. We have been colleagues for many years, and it has been a pleasure working with him. Having already written another book with Rodney and knowing the quality of his writing, doing this book with him was an easy choice.

I would also like to thank my father, who introduced me to the world of electronics when I was still very young. His interest in digital electronics would prove to be the spark that set me on my path to a career in information technology. I remember so clearly the day he brought a book home called *Digital Computers Made Simple*. I read it and thought to myself how exciting it was that a machine could be built and programmed to do so many different things. I could see so many possibilities, and at that point I knew I was hooked.

My first exposure to computer technology started about the time the Intel 8008 became available. With my father's help, I was able to procure one of these early microprocessors, and it became the basis for the first microcomputer I ever built. Using these early technologies, I built and entered several computer systems in science-fair projects throughout my high-school years, eventually competing at the international level.

My corporate career began early when I started my first company while still in high school. Today I am the CEO of HealthWare Corporation, a company that provides information technology to the health-care field. I am also partners in several other businesses with interests ranging from technology to real estate.

I would like to thank my wife, Kathi, who has put up with the long hours that it took to write the book. She is a wonderful companion who understands me well and always provides support and encouragement for any endeavor I undertake.

I would also like to thank my mother, who encouraged me all those years in school when I was participating in science fairs. Thanks to my sister, who always tells me she gave up her computer technology genes so I could have them. And to the rest of my family, friends, and colleagues who supported me and my wife, Kathi, throughout the process of writing this book.

A very big thank you goes to HealthWare Corporation and all of its employees and customers. They have been instrumental in my career and have provided inspiration for writing the book.

Special thanks go to Bruce and Cindi Yarbrough, who welcomed me into their home when hurricanes forced us to move HealthWare's data center operations to Atlanta. Thank you also to my nephew Ethan for introducing me to gaming on Microsoft's Xbox while I was there.

Thanks also go to Apress and the wonderful people involved with this project for their important roles in getting this book to print: lead editor Tony Davis; project manager Sofia Marchant; technical reviewers Chris Rausch, Trish Middleton, and Tom Rizzo; copy editors Kim Wimpsett and Julie McNamee; production editor Laura Cheu; indexer Kevin Broccoli; proofreaders April Eddy and Linda Seifert; comp house Kinetic Publishing Services; and any others involved with their important roles in getting this book to print.

Special thanks go to Tom Rizzo, the SQL Server product manager at Microsoft, for answering many questions for us during the course of our writing and for putting us in contact with other resources within Microsoft when needed.

Walter J. Voytek II

Introduction

At its core, the process of designing reports hasn't changed substantially in the past 15 years. The report designer lays out report objects, which contain data from a known data source, in a design application such as Crystal Reports or Microsoft Access. He or she then tests report execution, verifies the accuracy of the results, and distributes the report to the target audience.

Sure, there are enough differences between design applications to mean that the designer must become familiar with each particular environment. However, there's enough crossover functionality to make this learning curve small. For example, the `SUM` function is the same in Crystal Reports as it is in Microsoft Access as it is in Structured Query Language (SQL).

With Microsoft SQL Server 2005 Reporting Services (referred to as SSRS throughout the book), there is, again, only a marginal difference in the way reports are designed from one graphical report design application to another. So, if you do have previous reporting experience, your learning curve for SSRS should be relatively shallow. This is especially true if you come from a .NET environment, because the report designer application for SSRS is Visual Studio 2005 or the application included with SQL Server 2005, Business Intelligence Development Studio (BIDS).

Having said all this, several differences set SSRS apart from other reporting solutions:

- It provides a standard reporting platform based on Report Definition Language (RDL), which is the XML schema that dictates the common structure of all SSRS reports. This allows for report creation from any third-party application that supports the RDL schema.
- SSRS is an integral part of the SQL Server 2005 release.
- SSRS offers features out of the box that in other products would be expensive additions to a basic deployment. These features include subscription services, report caching, report history, and scheduling of report execution.
- SSRS, being a Web-based solution, can be deployed across a variety of platforms.

This book was written in parallel with a real SSRS deployment for a health-care application, so it covers almost every design and deployment consideration for SSRS, always from the standpoint of how to get the job done effectively. You'll find step-by-step guides, practical tips, and best practices, along with code samples that you'll be able to modify and use in your own SSRS applications.

What This Book Covers

From designing reports and stored procedures in Chapters 2–4, to deployment, management, and security processes in Chapters 6–9, the book uses a standard real-world theme to show how we chose to work with SSRS. Throughout, you'll find tips and tricks that we discovered while working closely with SSRS. The book also covers extending SSRS functionality with custom code in Chapter 5. In addition, we will demonstrate almost all the enhancements that are included with the SQL Server 2005 version of SSRS, including multivalued parameters, interactive sorting, and an entire chapter devoted to the ad hoc Report Builder application.

The following is a chapter-by-chapter breakdown to give you a feel for what the book covers:

Chapter 1, "Introducing the Reporting Services Architecture": This chapter introduces SSRS and discusses some of the driving forces behind our company's adoption of this technology. We then take a detailed look at the component pieces of the SSRS architecture, including Report Manager, BIDS, and the SSRS report server and databases. We describe how these work together to provide an effective reporting solution. We will also highlight all the new features and enhancements to the latest version of SSRS for SQL Server 2005. We finish with installation and configuration instructions.

Chapter 2, "Report Authoring: Designing Efficient Queries": The foundation of any report is the SQL query that defines the report data. In this chapter, we examine the query development process and show how to build and test high-performance queries for business reports. We also show how to encapsulate such queries in parameterized stored procedures to benefit from precompilation and reuse.

Chapter 3, "Using Report Designer": This chapter explores BIDS in detail, demonstrating the use of all the major embedded elements of SSRS within that environment. It shows how to create data sources; how to add report parameters, filters, and expressions; and provides an in-depth look at the layout section for report design.

Chapter 4, "Building Reports": Having covered query and report design basics, we now walk you through the process of building a full business report, including interactive features such as document maps, hyperlinks, and bookmarks.

Chapter 5, "Using Custom .NET Code with Reports": This chapter shows you how to customize your reports using .NET code, either by embedding Visual Basic .NET code directly in your report or by using a custom .NET assembly. We discuss and demonstrate each technique and its pros and cons.

Chapter 6, "Rendering Reports from .NET Applications": This chapter shows how to control the rendering of your reports programmatically in a variety of supported formats, either via URL access or by using the Web services API.

Chapter 7, "Deploying Reports": SSRS provides several means of deploying reports: using the Report Manager interface, using VS .NET, using the `rs` command-line utility, or writing code using the Web services API. This chapter demonstrates and explains each of these techniques.

Chapter 8, "Managing Reports": This chapter examines the many facets of SSRS report management, including content management, performance monitoring, report execution auditing, and control. It shows how to perform each of these tasks effectively, using built-in tools such as Report Manager, the new Report Server Configuration Manager, and command-line utilities, as well as using custom .NET management tools.

Chapter 9, "Securing Reports": This chapter introduces several important components of SSRS security, namely, data encryption, authentication and user access, and report auditing. This chapter also shows how to use each of these components in a secure SSRS deployment.

Chapter 10, "Delivering Business Intelligence with SSRS": In our work, we found that by integrating SSRS with many of the other components of the business intelligence (BI) platform, we were able to provide all the necessary information to our employees wherever they were and whenever they needed it, thus dramatically improving our overall business strategy. In this chapter, we demonstrate how we set about integrating SSRS with BI components such as CRM, SharePoint Portal Server, and Analysis Services.

Chapter 11, "Performing Ad Hoc Reporting Using Report Builder": In the final chapter, we will demonstrate the much anticipated Report Builder application, a Web-based report design tool for end users to build their own reports. The reports are created using report models as data sources. We will show how to build and deploy a report model and tap into it with Report Builder.

In each chapter, we've tried to touch on every aspect of SSRS in enough detail to allow you to translate the concepts into your own applications. Our intention was to provide truly practical, useful information on every page and not to parrot material that's adequately covered in Books Online (BOL). To that end, concepts such as cascading parameters and designing reports with hierarchical data using the `LEVEL` function aren't covered, because you can find adequate explanations and working examples in BOL.

We believe this book will serve as both an introduction and a step-by-step guide through many common tasks associated with SSRS, while also offering concepts and solutions that we've been developing ourselves for our own applications.

Who This Book Is For

We coauthored the book with the intention of demonstrating how to use SSRS from multiple vantage points. As a data analyst and engineer, Rodney goes through the report design and deployment processes using standard SSRS tools such as Report Designer and Report Manager. As a .NET developer, Jim takes on the role of showing how other developers can extend SSRS by creating custom Windows Forms applications, as he explains the SSRS programming model.

Source Code

In this book, we use a subset of a real database designed for a health-care application that we developed. You can find that prepopulated database (which we named `Pro_SSRS`, for the book), the data mart database and cube file used in Chapter 10, the completed RDL files, queries, stored procedures, and .NET application projects, as well as full installation instructions, in the Source Code section of the Apress Web site (`http://www.apress.com`).

CHAPTER 1

Introducing the Reporting Services Architecture

When Microsoft announced in 2003 that it was going to release SQL Server Reporting Services (SSRS) as a SQL Server 2000 add-on, a frenzy of excitement ensued. The product was originally slated for release with SQL Server 2005, so the early release was a welcome event for many. Our software development company decided to embrace SSRS early on and was fortunate to work with Microsoft during the beta phases. In January 2004, the month SSRS was released to manufacturing (RTM), we deployed it immediately. We intended to migrate all of our existing reports (which had been developed on as many as five reporting applications and platforms over the past ten years) to SSRS. We can sum up the reason for the seemingly rapid decision in one word: *standardization*.

Just as Microsoft wanted to create an industry standard with Report Definition Language (RDL), the Extensible Markup Language (XML) schema that dictates the common structure of all SSRS reports, we wanted to provide a standard reporting solution to our customers. Even in the first version of the product, SSRS delivered almost all the features we needed. Thanks to its extensibility via SSRS's Web service, we could programmatically add other features that weren't already directly supported. In addition, Microsoft was committed to enhancing SSRS over time. Even prior to SSRS for SQL Server 2005 (SS2005), Microsoft provided valuable enhancements such as client-side printing in service pack releases.

That brings us to present day, to the SSRS enhancements that have been incorporated into the long-awaited release of SQL Server 2005. SSRS has taken its place as a key component in the latest release of SQL Server and can no longer be thought of as just an add-on. The new features in SSRS are extensive (we provide an overview of these features shortly), and in most cases the incubus for their inclusion in the SSRS for SQL Server 2005 release was direct user feedback. Throughout the book, we will demonstrate each of these new features as we show how to build reports and applications. Furthermore, we'll explore how SSRS integrates with other new features in SSRS of SQL Server 2005 and how you can utilize these features to build comprehensive and effective business intelligence (BI) and Web reporting solutions.

Understanding the Benefits of SSRS

The decision of our company to migrate immediately to SSRS was based on the following perceived benefits for the company and for our customers:

Standard platform: In addition to providing a standard realized with the RDL, our development teams had been using Visual Studio .NET (VS .NET) as their main development environment. Because SSRS reports were currently developed within this platform, we wouldn't need to purchase additional development software. Our clients would need to purchase only a low-cost edition of a designer—Visual Basic (VB) .NET, for example—to gain the benefit of developing custom reports themselves. In SQL Server 2005, Business Intelligence Development Studio (BIDS) is included as a free, alternative report designer. Because it is based on Visual Studio 2005 (VS 2005), report designers who learn to design reports with BIDS can move to the full VS 2005 environment anytime with no additional training.

Cost: SSRS is an integral part of SQL Server 2005 and is available in many editions, from Express to Enterprise. When you purchase SQL Server, you get SSRS as well.

Web-enabled: Because SSRS is a Web-based reporting solution, a single deployed report is accessible to a variety of clients, from the browser to custom Windows Forms. Also, because reports are accessed via Hypertext Transfer Protocol (HTTP) or HTTP Secure (HTTPS), you can view reports from any location that has access to the SSRS Web server, which no longer requires reports to be installed locally with heavy client applications.

Customizable: SSRS provides a .NET Web service as a front end and as such can be accessed programmatically to extend the delivery of reports beyond the browser. As .NET programmers, we knew we would want to build custom applications to render reports where we could control the look and feel of the report viewer. We show one such application in Chapter 6, which covers report rendering.

Subscriptions: Having the ability to deliver reports through e-mail or a file share and processed during off-peak hours, which was offered with SSRS subscription abilities, was a huge advantage for our company and our clients. We show how to set up two different kinds of subscriptions, standard and data-driven, in Chapter 8.

As you'll see, SSRS is a full reporting solution that encompasses many levels of professional expertise, from report design to database administration. In many organizations, especially small- to medium-sized ones, information technology (IT) professionals are asked to perform many jobs. They write a query and design a report in the morning, perform database backups or restores in the afternoon, and update all the systems before heading home.

Fortunately, during external deployment of SSRS to our clients and internal deployment for my software development company, I (Rodney) have worn each of these hats on a day-to-day basis. I have been entrenched in every deployment phase. By developing efficient stored procedures, designing reports, testing security, and maintaining deployed reports as a content manager, I have witnessed the day-to-day operation of SSRS from many perspectives.

In addition to those roles, I have also been responsible for our company's overall strategy for building solutions to analyze and transform the data that's gathered through both our own and other third-party applications. To that end, an essential part of my job was integrating SSRS into the overall BI strategy that incorporated the following:

- Disparate data sources such as Analysis Services Cubes and SQL Server relational databases
- Applications and tools such as Microsoft Excel and Business Scorecards
- Document management systems such as Microsoft SharePoint Portal Server

We'll dive into the details of such integration projects in Chapter 10, which is devoted to BI.

SSRS represents another world—a world that an administrator who uses standard management tools doesn't typically witness. That is the world of the software developer who can extend and control SSRS programmatically, building custom report viewers and deployment applications. In this book, as you work through each step of building a reporting solution for health-care professionals, we'll demonstrate how an administrator can accomplish the task with built-in tools, as well as how a developer can create an application to provide enhanced functionality.

SSRS IN CONTEXT: GREEN BAR, ANYONE?

Before we begin our breakdown of the overall SSRS platform, I (Rodney) will share a personal experience that illustrates one of the many challenges that SSRS addresses. That is, the story shows that creating an environment where the method in which the data is delivered to users is often as crucial as the data itself. Users want easy and fast access to data in an intuitive but powerful interface. SSRS overcomes this challenge of changing the way users work by delivering reports in applications that are already familiar to most users: browsers and e-mail clients.

Jumping back in time a few years—well, 12 years—when I started down the path of what is now described correctly as IT, I took a job as an intern in the government sector. I should have known by the *Data Processing Center* banner over the door to my interviewer's office that I wasn't exactly stepping into the modern digital age. I was offered the lowly position of mainframe computer operator, which I took eagerly despite that I knew I would be eating boiled eggs and tomato soup for the foreseeable future. On the first day, I was introduced to two assemblages of technology that my father, who also worked in data processing (DP), introduced me to in the early 1980s: a vault full of reel-to-reel magnetic tapes and box after box of green bar paper. In time I came to both appreciate and loathe the rote task of, every night, printing thousands of pages of reports that I knew would only be scanned by a few people and then discarded. I say that I appreciated the task because every so often I would be visited by a programmer who wrote one of the reports. We'd talk about the time it took to write the report, why he constantly had to update the reports, and who was asking for the updates (typically a high official). We would also commiserate about the fact that generating such a report each night was a complete waste of valuable resources. I could only hope he meant me.

One day I heard a rumor that my beloved Data Processing Center was going to be absorbed by another government body. I learned, as many did, that sweeping changes would affect my position. New supervisors came in and surveyed the inherited archaic technology landscape. What happened was astounding in many regards. The banner over my former boss's door was the first to be altered; with the stroke of a paintbrush we were now *Information Resources*. The new regime didn't think "computer operator" was a good title for me anymore. In less than the time it takes to print 3,000 checks, I became a "data specialist," and I could now eat spaghetti with real meat sauce.

I bided my time, awaiting a new system that would mean I could take the reins in my new administrative position and stop hauling the green bar. A new system was duly purchased, and I was elated. Finally, I thought, they'll bring in a modern networked system that will have online report delivery technologies. On the day the hardware was delivered, I looked in awe at my—technically, their—new IBM RS6000. Fortunately for everyone except me, the new printer that was delivered accepted the same green bar paper that we had stockpiled for years to come!

This story demonstrates that often the benefits of new technologies go unrealized because of the habitual nature of the workers who are forced to adopt that new technology. Users who relied on the Data Processing Center were used to receiving their 200-page reports each morning and tossing them by noon. If the reports weren't discarded, they were bound, bundled, and hauled to the basement for future reference. It had been done that way for years. It seemed that only the people who appreciated the work that went into the reports—the programmers and computer operators—understood the ridiculousness of the practice.

In the ensuing years, I had a number of positions, all of which involved delivering data in myriad reporting technologies, using a plethora of data stores. One day it was scrubbing data from Indexed Sequential Access Method (ISAM) files; the next day saw me pulling data from a Unix-based Oracle database. With the blessing of Open Database Connectivity (ODBC), data was accessible in almost any format.

Eventually I began work with Microsoft SQL Server 6.5 and have remained there through versions 7.0 and 2000. Over the latter years, Microsoft released a variety of applications that made my job significantly easier while offering much in the way of data delivery. A notable example was the introduction of Online Analytical Processing (OLAP) Services in 7.0, which became Analysis Services in 2000. However, one item that always seemed just out of reach was a client application that could be used to effectively deliver the data contained within these new technologies. Sure, Excel could tap into OLAP cubes, and Data Analyzer was a promising addition, but these were expensive applications that required local installations. Surely a Web-enabled reporting application would be available soon, right?

This brings me to the present, where SSRS has been unveiled and is holding its own as a prime contender in the reporting market space. SSRS has been available as an add-on to SQL Server 2000 for more than a year now. It met or exceeded my expectations as a version 1 product in most areas. Yet, from reading the public newsgroups and through my own experience, it was obvious that Microsoft needed to address a number of shortcomings. A couple of the SSRS enhancements, such as client-side printing functionality and SharePoint Web Parts for Reporting Services, made their way into the two service packs that have been released for SSRS for SQL Server 2000. However, many other features are specific to SQL Server 2005. In the following two sections, we'll cover first the new SSRS features and then the enhancements to Microsoft's BI platform, with which SSRS can be fully integrated (as we will demonstrate throughout the book).

SQL Server 2005 Reporting Services Enhancements

The following are the most significant enhancements made to the SSRS technology that have emerged since version 1.

Report Builder/Data Modeler

The Report Builder application, a new feature to SSRS for SQL Server 2005, is a local, ad hoc, report-designing application that is intended to be used more by report consumers than by report developers. The business logic and underlying data structures are created as a data model by an administrator who is familiar with the source data. With the Report Builder application, the user can create and publish reports based on available models. Chapter 11 covers how to build and deploy a data model as well as create reports with the Report Builder application.

Multivalue Parameters

SSRS 2000 supported report input parameters, allowing for the autogeneration of report parameters from query or stored procedures parameters, drop-down selectable data values, and default values. However, it soon became evident that the inability to select more than one value at a time to pass into the report was a serious limitation. What if you needed to see, for example, 2003 and 2004 data from a list of available values such as "2002,2003,2004"? In SSRS for SQL Server 2000, you could see 2003 *or* 2004 *or* all the years, but 2003 *and* 2004 was not an available option. SSRS for SQL Server 2005 addresses this limitation. We will cover multivalued parameters in Chapter 4.

Enhanced Expression Builder

Expressions are the core components of any SSRS report. With expressions it is possible to selectively control not only the content but also the desired behavior of report. Knowing this, Microsoft has included an enhanced expression editor in SSRS for SQL Server 2005, replacing the limited version in the previous release. The expression builder in SSRS for SQL Server 2000 was little more than a textbox for code entry. Now the expression builder includes a categorized list of available functions, such as String and DateTime functions, as well a built-in IntelliSense that validates the expression as it is being typed.

Graphical MDX Query Builder

Multidimensional Expressions (MDX), used to query OLAP data cubes, is a robust language that when used effectively can derive information that would be impractical to do with Transact SQL (T-SQL). However, MDX is a complex language with strict syntactical requirements, and thus MDX queries can be tricky to formulate. In SSRS for SQL Server 2000, MDX was supported, but the queries could not be created graphically. SSRS for SQL Server 2005 includes a graphical MDX query builder that you will use in Chapter 10 when designing your Analysis Services application.

Embeddable SSRS Controls

The ability to embed controls in custom applications makes it easier for developers to integrate SSRS into their projects. SQL Server 2005 includes freely distributable controls that you can use for Windows Forms development and ASP.NET Web Forms development. These controls provide additional benefits to developers, such as the ability to render reports while disconnected from the SSRS. We will cover SSRS controls in Chapter 6.

Rich Client Printing

SSRS has always had the capability to deliver reports in a variety of formats, but not all of them were print-ready. Reports rendered in Hypertext Markup Language (HTML), for example, provided interactive features such as drilling down, but printing large reports rendered in the browser was not printer-friendly. Often, sections of the report would be truncated or paginated incorrectly. You could create delivery extensions with SSRS for SQL Server 2000 to print directly to a client printer. With SSRS for SQL Server 2005, this functionality is built in.

Interactive Sorting

The ability of users to sort report data after rendering provides another level of interactivity that, potentially, mitigates the need to deploy multiple reports. SSRS for SQL Server 2005 allows report developers to define interactive sorting, which you will explore in Chapter 4.

SSRS and Business Intelligence

Because SSRS is but one component of Microsoft's BI platform, we'll now cover other new features and enhancements to SQL Server 2005 that will form an integral part of your overall reporting solution.

Business Intelligence Development Studio (BIDS)

BIDS is a limited version of Visual Studio 2005 that is included with the SQL Server 2005 base installation. With BIDS, developers can create entire projects for each of the supported components of SQL Server 2005, including SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), and of course SSRS. We will use BIDS throughout the book to show how to design and deploy SSRS reports and Analysis Services projects.

SQL Server Management Studio (SSMS)

With the new SQL Server Management Studio (SSMS), Microsoft has taken a big step toward consolidating within one environment many of the tools that in previous versions of SQL Server would have been executed individually. SSMS replaces Enterprise Manager and Query Analyzer, offering a much more elaborate set of tools for creating and managing SQL Server objects and queries. In addition to managing SQL Server and Analysis Services servers, administrators can use SSMS to manage instances of their SSRS reporting servers. In the previous version of SSRS, this level of control was available graphically through the browser-based Report Manager or a custom application.

We will show how to use both SSMS and Report Manager throughout the book for different tasks. We will show how to use SSMS, for example, to test query performance, a task done previously with Query Analyzer. In addition, we will show you how to use the browser-based Report Manager to view published reports, set security permissions, and create subscriptions. Both applications share functionality for managing SSRS; however, Report Manager is often preferable to SSMS because it does not require a local installation. You can access Report Manager from a browser anywhere on your network. You would need to have access to the installed SQL Server 2005 client tools in order to use SSMS.

SharePoint Web Parts

SharePoint, either in Microsoft SharePoint Portal Server (SPS) or in Windows SharePoint Services (WSS), has become one of Microsoft's most successful products. SSRS for SQL Server 2005 provides tighter integration with SharePoint by way of Web Parts that connect directly to SSRS reports on the report server. Chapter 10 covers how to incorporate these controls into a BI application.

Exploring the SSRS Architecture

You've probably heard the expression that the devil is in the details. You'll be drilling into those details throughout the book, right down to the data packets that SSRS constructs, as you explore each aspect of SSRS from design to security. For now, let's pull back to a broader vantage point—the 10,000-foot view—and look at the three main components that work together to make SSRS a true multitier application: the client, the report server, and the SQL Server report databases. Figure 1-1 shows the conceptual breakdown of the three component pieces.

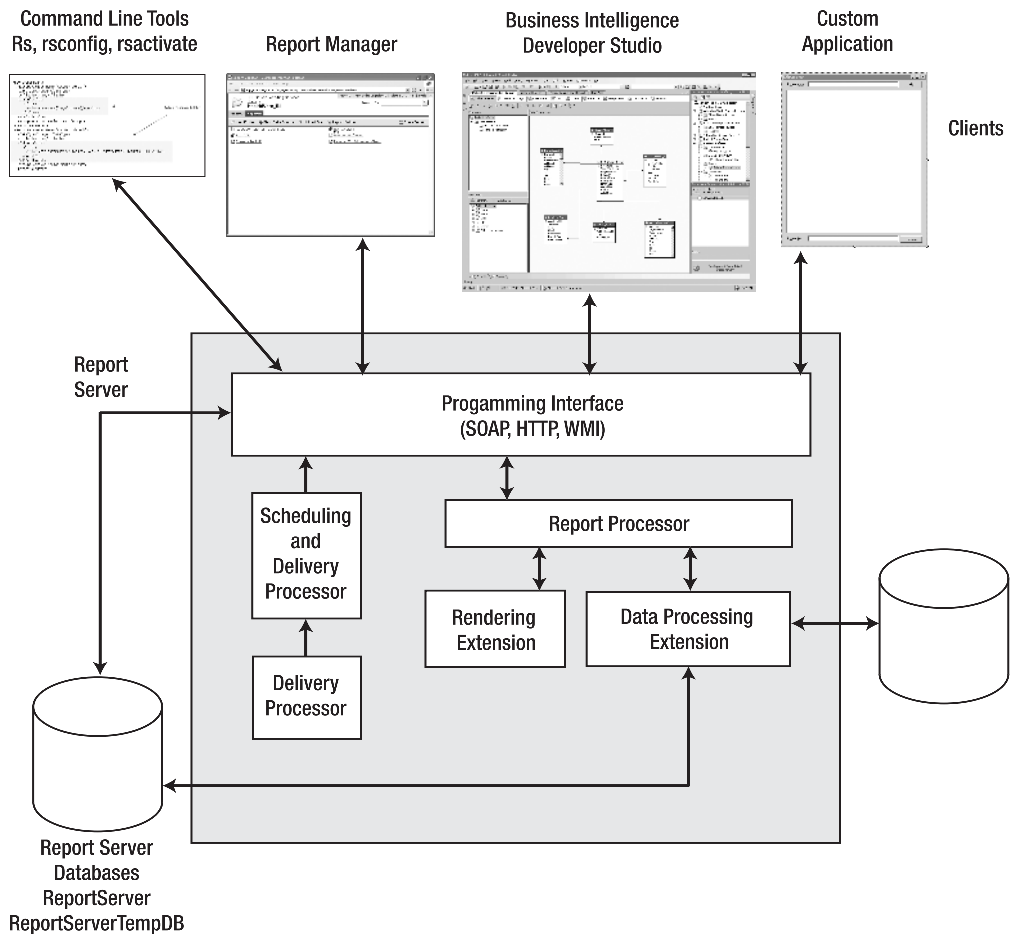


Figure 1-1. *SSRS components*

Here, the data source and the SSRS databases, `ReportServer` and `ReportServerTempDB`, are separate entities; the data source is the origin of the data that will populate the reports; and the report server databases store information about the reports. Both the data source and the report server databases can physically be located on the same SQL Server, assuming the data source is a SQL Server database. The data source can be any supported data provider, such as SQL Server, Oracle, Lightweight Directory Access Protocol (LDAP), or Analysis Services. It's possible to configure a single server to act as both the SSRS report server web service and report server database as well as the data source server. However, this isn't recommended unless you have a small user base. We'll show how to monitor the performance of the SSRS configuration and build a small Web farm, post-installation, in Chapter 8.

SSRS Databases

The SSRS installation creates two databases:

`ReportServer`: This is the primary database that stores all the information about reports that was originally provided from the RDL files used to create and publish the reports to the `ReportServer` database. In addition to report properties (such as data sources) and report parameters, `ReportServer` also stores folder hierarchy and report execution log information.

`ReportServerTempDB`: This database houses cached copies of reports that you can use to increase performance for many simultaneous users. By caching reports using a nonvolatile storage mechanism, you make sure they remain available to users even if the report server is restarted.

Database administrators can use standard tools to back up and restore these two databases. An additional database might be added after the initial installation of SSRS: the `RSExecutionLog` database. This database stores more discernable information about report execution, such as the user who ran the report, the time of execution, and performance statistics. We'll cover creating the `RSExecutionLog` database and discuss report execution logging in detail in Chapter 8.

The SSRS Report Server

The SSRS report server plays the most important role in the SSRS model. Working in the middle, it's responsible for every client request to render a report or to perform a management request, such as creating a subscription. You can break down the report server into several subcomponents by their function:

- Programming interface
- Report processing
- Data processing
- Report rendering
- Report scheduling and delivery

SSRS Web Service Interface

The programming interface, exposed as .NET Web service application programming interfaces (APIs) and uniform resource locator (URL) access methods, handles all incoming requests from clients, whether the request is a report request or a management request. Depending on the type of request, the programming interface either processes it directly by accessing the `ReportServer` database or passes it off to another component for further processing. If the request is for an on-demand report or a snapshot, the Web service passes it to the Report Processor before delivering the completed request to the client or storing it in the `ReportServer` database.

■**Note** On-demand reports are ones that are rendered and delivered directly to the client, while snapshots are reports that are processed at a point in time and delivered to the client through e-mail, to the client through file shares, or (if configured) directly to a printer.

The Report Processor

The Report Processor component is responsible for all report requests. Like the programming interface, it communicates directly with the `ReportServer` database to receive the report definition information that it then uses to combine with the data returned from the data source, which is accessed via one of the data processing extensions.

Data Processing

SSRS supports four data processing extensions to connect to data sources. These are SQL Server, Oracle, OLE DB, and ODBC. When the data processing component receives the request from the Report Processor, it initiates a connection to the data source and passes it the source query. Data is returned and sent back to the Report Processor, which then combines the elements of the report with the data returned from the Data Processor extension.

Report Rendering

The combined report and data is handed off to the rendering extension component to be turned into one of several supported formats, based on the rendering type specified by the client (we cover rendering in depth in Chapter 6):

- *HTML*: Default rendering format, supporting HTML versions 4.0 and 3.2.
- *Portable Document Format (PDF)*: Format used to produce print-ready reports using Adobe Acrobat Reader. SSRS doesn't require that you have an Adobe license to render in PDF, which is a great benefit to customers. All you need is a PDF reader.
- *HTML using the Office Web Components (OWC)*: Can be used for rendering reports that are designed to take advantage of the extended features of OWC, such as a pivot table or chart. HTML for OWC has been depreciated in SSRS 2005 and needs to be enabled post installation. We show you how to do this in Chapter 8.
- *Excel 2002 and 2003*: Service Pack 1 of SSRS supports Excel 97 and later.
- *XML*: Other applications or services can use reports that are exported to XML.

- *Comma-separated values (CSVs)*: By rendering to a CSV file, you can further process the report by importing it into other CSV-supported applications such as Microsoft Excel.
- *MIME HTML (MHTML)*: You can use this format, also known as a *Web archive*, to deliver reports directly in e-mail or to deliver them for storage, because the report contents, including images, are embedded within a single file.
- *Tagged Image File Format (TIFF)*: Rendering image files using TIFF guarantees a standard view of the report, as it's processed the same way for all users despite their browser settings or versions.

Scheduling and Delivery

If the request from the client requires a schedule or delivery extension, such as a snapshot or subscription, the programming interface calls the Scheduling and Delivery Processor to handle the request. You can generate and deliver report snapshots based on a user-defined or shared schedule to one of two supported delivery extensions: an e-mail or a file share. Note that SSRS uses the SQL Server Agent to create the scheduled job. If the SQL Server Agent isn't running, the job won't execute. We'll cover creating subscriptions and snapshots based on shared schedules in Chapter 8.

Client Applications

SSRS includes several client applications that use the SSRS programming interface, namely, its Web service APIs, along with URL access methods to provide front-end tools for users to access both SSRS reports and configuration tools. These tools provide report server management, security implementation, and report-rendering functionality. The tools are as follows:

- *Report Manager*: This browser-based application ships with SSRS and provides a graphical interface for users who need to view or print reports or to manage report objects for their workgroups or departments. We cover Report Manager in detail in Chapter 8, which covers managing SSRS.
- *BIDS*: This tool provides an integrated environment for developing SSRS reports. We introduce BIDS in Chapter 3 and step through building reports in this environment in Chapter 4.
- *Command-line utilities*: You can use several command-line tools to configure and manage the SSRS environment, including `rs`, `rsconfig`, `RSKeyMgmt`.
- *Custom clients*: These VB .NET Windows Forms and Web applications call the SSRS Web service to perform such tasks as rendering reports and managing report objects. SSRS includes sample application projects that you can compile and run to extend the functionality provided by the main tools listed earlier. In Chapters 6 and 7 we show how to develop your own custom applications: a report viewer and a report publisher.
- *Reporting Services Configuration Manager*: Previously, SSRS relied on command-line utilities or on direct modification of config files for changing many of the properties of the SSRS installation. SSRS for SQL Server 2005 includes a new tool designed specifically to change many of these properties in a graphical environment.

When thinking of a Web-based application, the natural inclination is to think *Web browser*. Even though other front-end tools, such as SSMS and BIDS, connect to the report server, a Web browser plays an important role in providing the graphical interface for users who view or print reports or remotely manage the report server for their workgroups or departments.

Report Manager

Within Report Manager, users can render reports, create report subscriptions, modify the properties of report objects, and configure security, as well as perform a host of other tasks. Users can access Report Manager by simply opening their Web browser and navigating to a URL of the form `http://`*`Servername`*`/Reports`. Figure 1-2 shows Report Manager in action, with a listing of reports in a folder deployed specifically for clinicians.

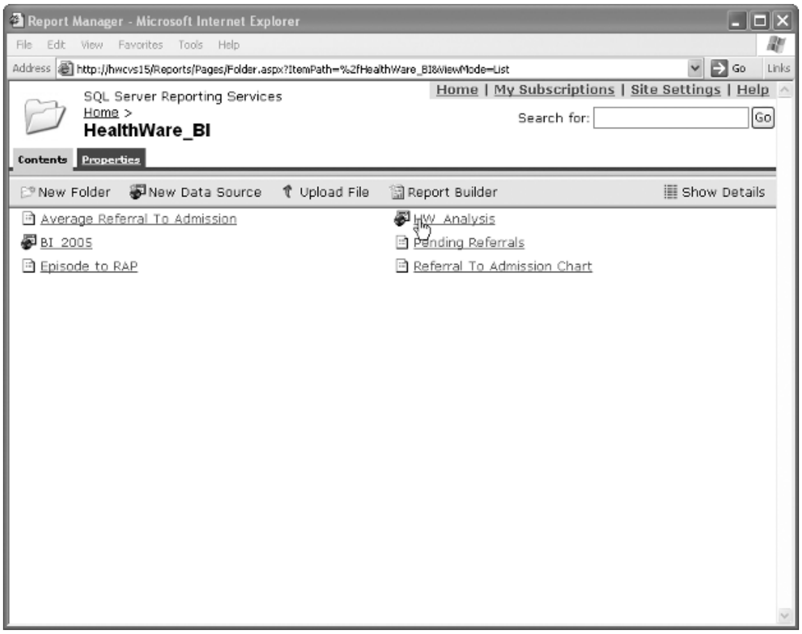

Report Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://hwcvs15/Reports/Pages/Folder.aspx?ItemPath=%2fHealthWare_BI&ViewMode=List
Go
Links
SQL Server Reporting Services
Home >
HealthWare_BI
Home | My Subscriptions | Site Settings | Help
Search for:
Go
Contents
Properties
New Folder
New Data Source
Upload File
Report Builder
Show Details
Average Referral To Admission
BI_2005
Episode to RAP
HW_Analysis
Pending Referrals
Referral To Admission Chart

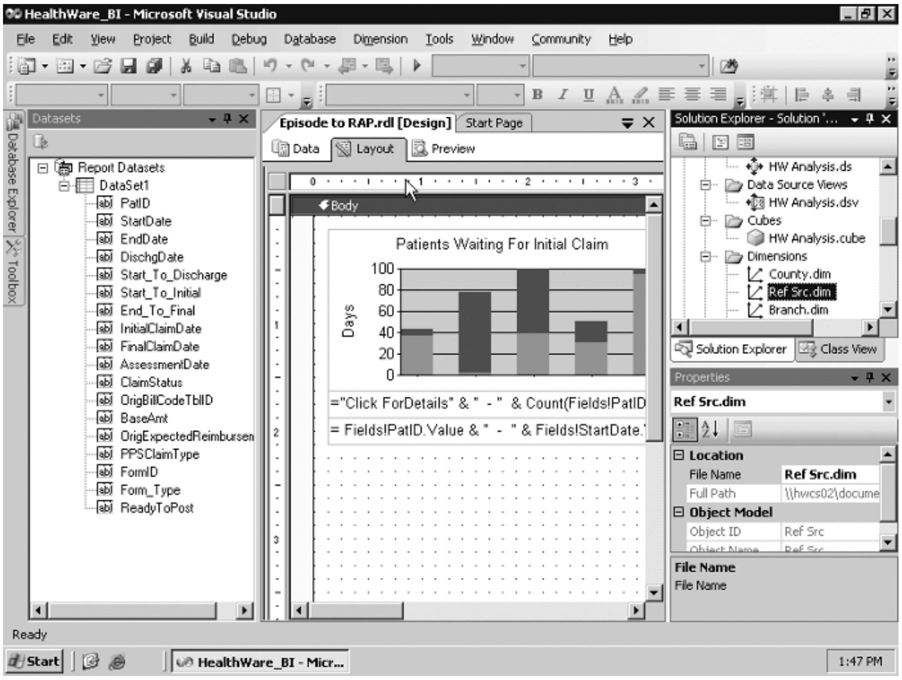
Figure 1-2. *The Web-based Report Manager application*

Business Intelligence Development Studio (BIDS)

The browser is only one of several clients that can use the SSRS Web service. In fact, BIDS is a client when it interacts with the Web service to deploy reports and data sources. BIDS offers a graphical design environment that report developers use to produce the RDL files that SSRS uses for deploying and rendering reports.

■**Note** Because RDL is a defined standard, you can use any design application that supports the creation of RDL files. Other third-party report designers are available, and many more are forthcoming.

By defining the base URL and folder name in a BIDS report project, you can deploy the RDL files that are created directly to the report server while in design mode. The base URL is of the form `http://`*`Servername`*`/ReportServer`. We'll cover the entire BIDS design environment in Chapter 3, including most available report objects. We'll also describe the RDL schema that defines every aspect of an SSRS report. Figure 1-3 shows the BIDS design environment, also called an integrated development environment (IDE), with a report loaded in design mode.



```
="Click ForDetails" & " - " & Count(Fields!PatID
= Fields!PatID.Value & " - " & Fields!StartDate.
```

File Name: Ref Src.dim
Full Path: \\hwcs02\docume
Object ID: Ref Src

File Name
File Name

Figure 1-3. *BIDS environment*

Command-Line Utilities

In addition to graphical applications such as BIDS and SSMS, SSRS provides several command-line utilities that are considered Web service clients. The tools have the added benefit of being automated by using built-in task scheduling in Windows. SSRS includes four main command-line utilities:

- `rs`: Processes report services script (RSS) files that are written with VB .NET code. Because you can access the full SSRS API via the code in the script, all SSRS Web service methods are available.

- `rsconfig`: Configures the authentication methods and credentials for how SSRS connects to the `ReportServer` database. `rsconfig` also sets the unattended SSRS execution credentials for SSRS.

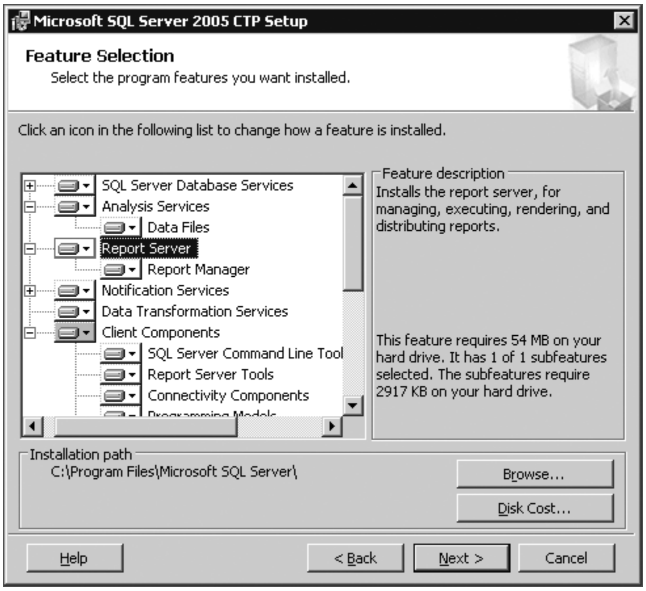
- `RSKeyMgmt`: Manages the encryption keys that SSRS uses to store sensitive data securely, such as authentication credentials. Chapter 9 covers how to use `rskeymgmt`.

Custom Clients

The final types of clients are those custom designed to access the SSRS Web services. We've built several such applications for our own company, such as a report viewer and report publisher. Third-party commercial applications exist that provide extended functionality. Other clients, such as the new Report Builder application, is a good example of building not just a report-rendering form but an entire design application that connects directly to the report server and can be installed using standard ClickOnce technologies from within the browser.

Installing and Configuring

You can install SSRS—like Analysis Services, Integration Services, and Notification Services—as part of the main SQL Server 2005 installation. When installing SSRS, you can choose which components to install and also specify the initial configuration settings. You can see the components available within the SSRS install portion of SQL Server 2005 in Figure 1-4; these include both server and client components.


Microsoft SQL Server 2005 CTP Setup

Feature Selection

Select the program features you want installed.

Click an icon in the following list to change how a feature is installed.

Feature description

Installs the report server, for managing, executing, rendering, and distributing reports.

This feature requires 54 MB on your hard drive. It has 1 of 1 subfeatures selected. The subfeatures require 2917 KB on your hard drive.

Installation path

C:\Program Files\Microsoft SQL Server\

Figure 1-4. *Installation of SSRS components*

Server components include the report server Web service, `ReportServer` databases, and Report Manager. When installing the server components, you may have the option of configuring the installation to connect to an already existing SSRS database. By choosing this option, the instance of SSRS you're installing joins a Web farm of other SSRS servers, all using the same `ReportServer` database.

Note The standard edition of SSRS doesn't provide support for setting up Web farms. Throughout the book, we'll be using the Developer edition of SSRS, which provides support for Web farms and for another enterprise feature: data-driven subscriptions.

Client components include the administrative command-line tools mentioned previously, such as `rs` and `rsconfig`, as well as documentation, samples and the Reporting Services Configuration Manager.

As noted, the install process also allows you to set your initial SSRS configuration. For example, you can do the following:

- You can choose security settings, such as whether the report server will use HTTP, use HTTPS, or use both.
- You configure a Simple Mail Transfer Protocol (SMTP) mail server to handle the delivery of subscriptions.

After you install SSRS, you can modify the configuration settings you chose during the install in a few ways. For example, after reviewing performance data, you might decide that the report server needs to connect to an existing Web farm. You can perform this task using the `rsconfig` utility or using the graphical Report Services Configuration Manager.

You can reconfigure the security settings or the mail server by directly modifying the `RSReportServer.config` file. We'll cover using these tools, modifying the configuration file settings, and gathering performance measures in Chapters 8 and 9.

Deploying SSRS Securely

Security ranks as one of the highest priorities for businesses today. Providing customers and employees with a secure and reliable computing environment is not only good practice but in many cases it's a requirement, mandated by stringent federal regulations. In our case, this meant adherence to the Health Insurance Portability and Accountability Act (HIPAA), which requires policies and procedures to be in place to guarantee that confidential patient information is securely transmitted and accessible only by those with the authority to view it. To that end, we have to ensure that the data we transmit over a network connection, especially the Internet, is encrypted at its source.

SSRS is a role-based application that provides access to the objects it stores through the use of defined roles, such as content browsers who may only view reports and report data. The roles that SSRS provides are associated with Windows-based login accounts, so SSRS relies on Windows as its primary source of authentication. It is possible to extend the security model for SSRS to support other methods of authentication, such as forms-based authentication whereby users can log in with accounts maintained outside Windows to access the report server. Since SSRS has multiple authentication points—namely, at the report server level through IIS and the data-access level, SQL, or Windows authentication—specific security risks exist when altering the default Windows roles-based security model. For one, IIS would need to be set up to allow anonymous access. Another is that SSRS can support only one security extension at a time. In other words, a single SSRS report server either can be extended to support a nondefault authentication model or can remain in default Windows authentication, but