Pro SQL Server 2008 Reporting Services

Copyright © 2008 by Rodney Landrum, Shawn McGehee, and Walter J. Voytek III

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-992-1

ISBN-10 (pbk): 1-59059-992-6

ISBN-13 (electronic): 978-1-4302-0652-1

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

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Jonathan Gennick
Technical Reviewer: Fabio Claudio Ferracchiati
Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Sofia Marchant
Copy Editor: Ami Knox
Associate Production Director: Kari Brooks-Copony
Production Editors: Laura Cheu, Kelly Gunther
Compositor: Susan Glinert Stevens
Proofreader: Liz Welch
Indexer: Broccoli Information Management
Artist: Dina Quan
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 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at 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 2008. 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 2008 push the technology one step further into becoming the reporting development environment of choice for programmers and designers, especially those who are already skilled with Visual Studio (VS) and Visual Basic .NET (VB .NET). Like with its predecessor, SSRS 2005, the long-awaited features for SSRS 2008 are mostly driven by direct feedback from the user community. Throughout the book, we will demonstrate each of these new features as we show how to design professional reports, applications, and solutions built on Microsoft's business intelligence (BI) initiatives. Throughout the book, we will focus on SSRS as a whole, building on features from each version from 2000 to 2008; however, we will notate which features are new to SSRS 2008.

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 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—VB .NET, for example—to gain the benefit of developing custom reports themselves. In SQL Server 2005, Microsoft included the Business Intelligence Development Studio (BIDS) as a free, alternative report designer. This free development environment is also available with SQL Server 2008, based on the next edition of Visual Studio, VS 2008. Report designers who learn to design reports with BIDS have the advantage of moving to the full version of Visual Studio anytime with no additional training.

Cost: SSRS is an integral part of SQL Server 2008 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 primarily 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 in the early evening.

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, and explore one of the key advancements of SSRS 2008, which is a tighter integration with SharePoint portal server, to the point that SSRS content can now be directly deployed, managed, and viewed all with SharePoint.

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, 15 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 knowing 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 is now in its third major revision with SQL Server 2008 and holding its own as a prime contender in the reporting market space. Through colleagues, newsgroups, and personal experience, I can see that SSRS has been adopted by many companies, often replacing existing reporting solutions altogether. I also realize that resistance to change is as strong as ever for many others. As someone who has been working with SSRS now for nearly 4 years, watching it mature, I have heard some of the reasons why managers and developers are reluctant to move to fully move to SSRS. One reason, a valid one, is that it is difficult to find report developers who have both business-specific knowledge, such as accounting, and proven SSRS development skills. Most business analysts designing reports, it is argued, are familiar with other reporting technologies, like Business Objects, so it is often easier for a company, despite the cost, to purchase the reporting solution that the ones who will be designing reports know how to use. Although this is not always the case—often companies will invest in retraining—this is one argument I have heard. Other reasons for not adopting SSRS are for specific features that do not exist or if they do, they are often lacking certain critical components. One of these limitations has been offline reporting, or the ability to view reports without being connected to the IIS Reporting Services Web front end. There are myriad other features that users have wanted. Microsoft does listen to these feature requests as is evidenced by the many updates that SSRS has seen over the years. SSRS 2008 will further deliver necessary features that will break down more barriers for user acceptance. We will cover these new features in the next section and in detail throughout the book.

SQL Server 2008 Reporting Services Enhancements

The following are the most significant enhancements made to the SSRS technology in SQL Server 2008.

Report Builder/Data Modeler

The Report Builder application, a feature introduced in SSRS 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. Report Builder 1.0, as well as the new Report Builder 2.0, based on Microsoft Ribbon technology, has been enhanced significantly in SSRS 2008, providing a richer development environment and additional content sources, such as Oracle and Analysis Services Cubes. Chapter 11 covers how to build and deploy a data model as well as create reports with the Report Builder 1.0 and Report Builder 2.0 applications.

Integrating SSRS 2008 with Microsoft Office SharePoint

While SharePoint integration has been available with the use of SharePoint controls in previous versions of SSRS, SSRS 2008 takes the integration several steps further. By using SSRS 2008 in SharePoint Integration Mode, it is possible to deploy, manage, and deliver reports and report objects, like data sources and models, all within the SharePoint environment. Entire dashboards or portals can be created using SSRS reports within SharePoint. In addition, the deployed reports inherit the native features of SharePoint, such as the ability to check in and check out reports, report change notification, and workflow capabilities. We will demonstrate this tighter integration with SharePoint in Chapter 10.

Introducing the Tablix Report Properties

As the name suggests, the new Tablix properties in SSRS 2008 are a combination of two existing controls, Table and Matrix. By combining these two report controls, developers now have a more flexible tool when creating reports with multiple columns and rows that blends the static nature of the Table control and the dynamic nature of the Matrix. Now it is possible to have a report that can accommodate multiple parallel rows and column members at each level, independent of each other but using the same aggregate calculations. In previous editions of this book, we provided workarounds to combining tables and matrices by embedding one within the other. We will now explore the true power of the new Tablix control properties for the List, Table, and Matrix controls in Chapter 4.

Enhanced Charting Visualizations

From the beginning, SSRS offered charts and visualizations natively in reports. These charts, while versatile, were somewhat limited in scope. Much or all of the functionality in the charting aspects of previous versions of SSRS could be easily duplicated in Microsoft Excel. In fact, the charting was almost identical. In SSRS 2008, there are several charting and graphical data visualization enhancements that are key to a sound BI reporting solution, which SSRS surely is a pivotal component of. New charting elements such as range, polar, radar, funnel, and pyramid are available as well as many new "gauges" delivered with the acquisition of Dundas reporting

controls for SSRS. We will explore several of these new visualizations as we incorporate them into reports in Chapter 4.

Enhanced Performance and Memory Management

The report engine in SSRS 2008 has been reengineered to lessen the memory footprint for reports at the server level, speeding delivery of reports to end-user applications. This enhancement also resolves contention with long-running, large reports with other smaller, non-memory-bound reports processing simultaneously.

Embeddable SSRS Controls

The ability to embed controls in custom applications makes it easier for developers to integrate SSRS into their projects. SQL Server 2008 includes updated 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 updated SSRS controls in Chapter 6.

HTML Text Formatting

Aside from the ability to export to Microsoft Word in SSRS 2008, text formatting is probably one of the most significant advancements to SSRS. In previous versions of SSRS, in-line formatting of textual content, such as a form letter, was not possible. SSRS 2008 report textboxes can now be formatted as you would in any word processor. The text formatting can combine both literal text and data source text for mail merge and template reports. We will demonstrate how to fully utilize this feature by creating a custom form letter style report in Chapter 4.

Exporting to Microsoft Word

Since the first version of SSRS, you could export any report to Microsoft Excel. While this was an important capability, not being able to export to other Microsoft Office formats, such as Word, was a limitation. Many reports that developers are asked to create require rich texts formats found in today's modern word processors. By combining the features of SSRS to design custom reports from multiple data sources with the richness of Microsoft Word formatting, SSRS 2008 has finally overcome a significant limitation.

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 2008 that will form an integral part of your overall reporting solution.

Business Intelligence Development Studio (BIDS)

BIDS is a limited version of Visual Studio 2008 that is included with the SQL Server 2008 base installation. With BIDS, developers can create entire projects for each of the supported components of SQL Server 2008, 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 SQL Server 2008, Microsoft continues to build on its management platform with 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.

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. 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 2008 client tools in order to use SSMS.

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.

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.

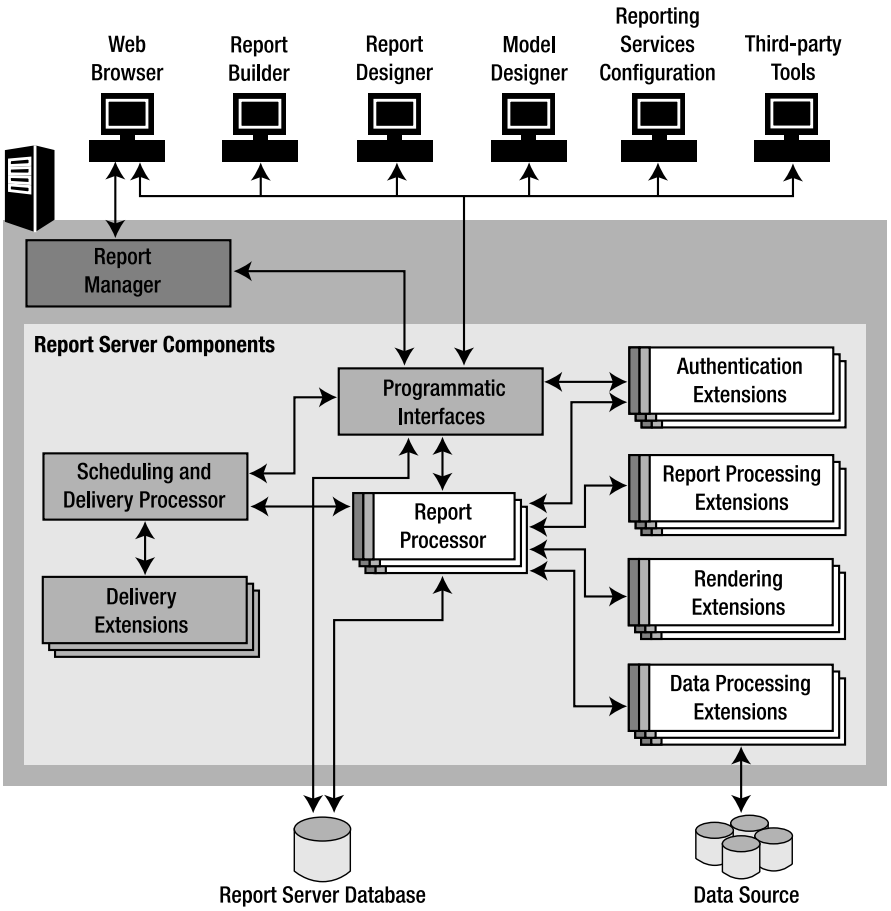

Figure 1-1. *SSRS components*

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
- Authentication Layer (new to SSRS 2008)
- 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 or via file shares, or (if configured) directly to a printer.

Authentication Layer

SSRS 2005 relied heavily on the authentication methodology of IIS, since SSRS and IIS were interdependent. SSRS 2008 is no longer tied to IIS; instead, it uses HTTP.SYS directly as well as SQL Server's native network components, and as such, SSRS's architecture has been redesigned to include its own authentication layer, which we will cover in Chapter 9.

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.
- *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.
- *Microsoft Word*: Standard Microsoft Word document export is now included in SSRS 2008.

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`, and `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*: SSRS for SQL Server 2008 includes an enhanced Reporting Services Configuration Manager designed specifically to change many of these properties in a graphical environment, including setting the SSRS environment up for offline or disconnected reporting.

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. They can use Report Manager to 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.

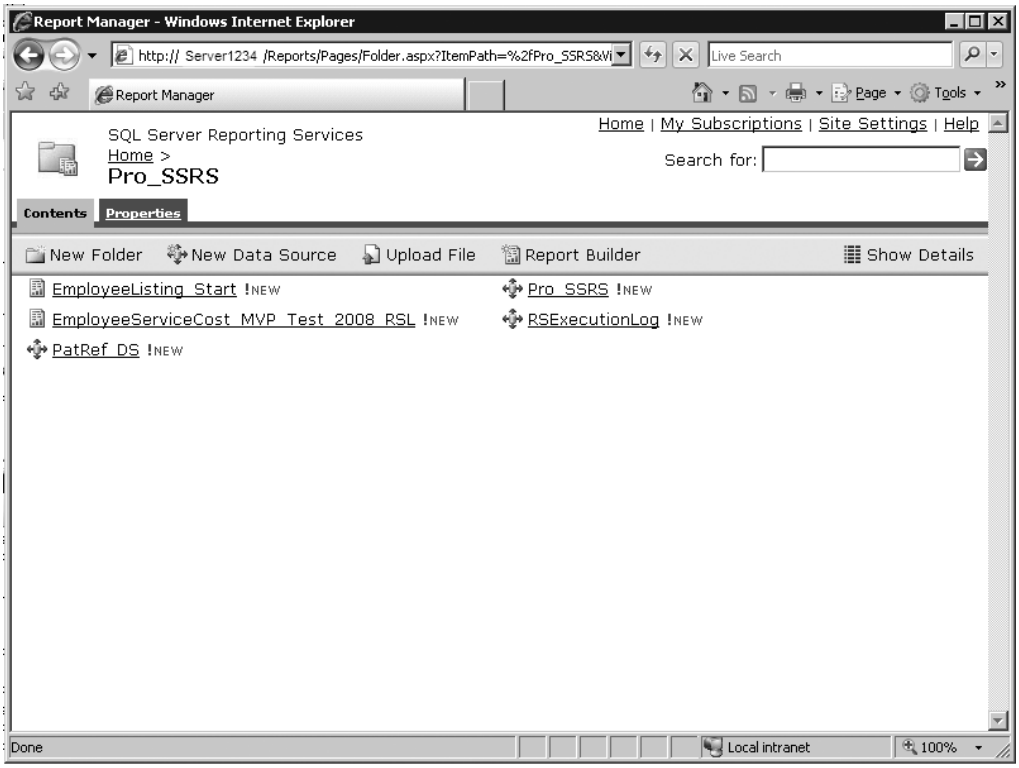

Figure 1-2. *The Web-based Report Manager application*

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.

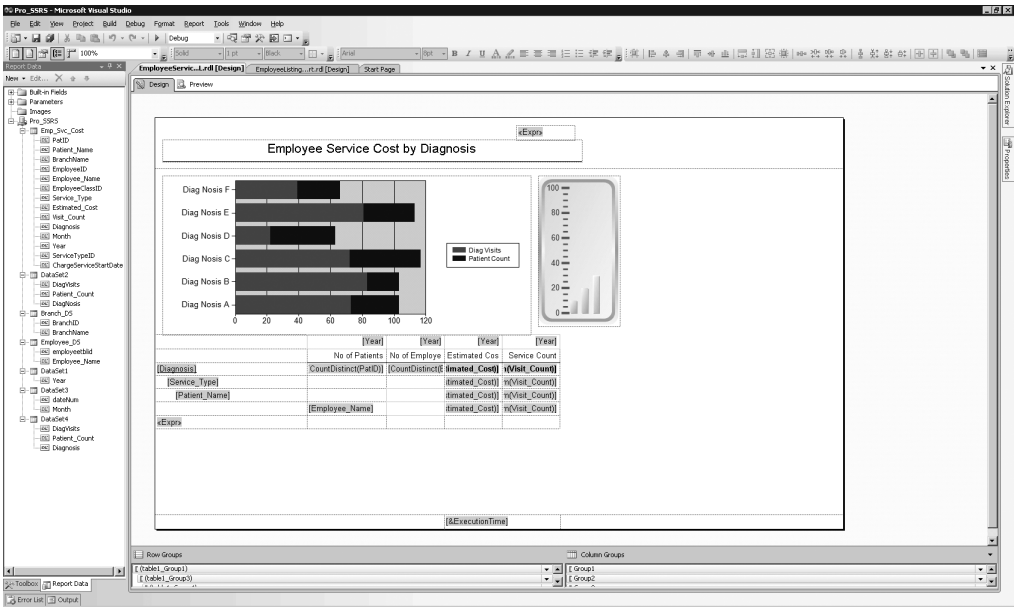

Figure 1-3. *BIDS environment*

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. One such client, 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 2008 installation. When installing SSRS, you can choose which components to install and also specify the initial configuration settings.

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.

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 Reporting 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 also in many cases 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 not both simultaneously. Depending on your level of need for custom security—say, for example, you need to deploy SSRS on an Internet-facing server, or your application already supports forms authentication, and it would be too difficult to work within the constraints of Windows authentication—then you might need to consider a custom security extension. Our needs were such that we could easily incorporate SSRS into an existing Windows authentication model.

In this book, we'll cover two deployment scenarios:

- Intranet deployment using Virtual Private Network (VPN) and firewall technologies to allow access to the SSRS report server
- Internet-hosted application that uses Terminal Services to connect securely to an SSRS report server

In Chapter 9, we'll walk you through securing the SSRS deployment models with technologies that provide the required encryption levels and user authentication. In addition to the two models that we will cover, we will also briefly discuss ways to integrate a forms-based authentication method that will allow clients to connect directly to SSRS via the Internet.

Summary

Having created and deployed numerous projects with SSRS for SQL Server 2005, we have been anxiously awaiting, along with the rest of the SQL Server community, the release of SQL Server 2008 and the promised enhancements to SSRS. As you work through the book, we will point out enhancements in SSRS 2008 where applicable, but our aim, as with the first editions of the book, is to show how to take advantage of advanced features, provide useful examples, and, mostly, put SSRS to work in a real-world environment where the user who will be working with the reports and applications that you deploy will have the final say on the solution's success.