CHAPTER 1

What Is Client-Side Reporting?

Client-side reporting is not a new phenomenon. No matter what platform you use for development, you need some way or other to produce reports. This chapter will explore the new client-side reporting features provided by Visual Studio (VS) 2005 and the forthcoming release of VS (Visual Studio 2008) and build a sound foundation for using them throughout this book. From here on, if you see a reference to "VS," I mean to say both VS 2005 and Visual Studio 2008.

In this chapter, the following topics will be covered:

- Characteristics of the client-side reporting architecture
- How client-side reporting supports various report users
- How clients can act as hosts for report delivery
- The three-step report-creation process
- Basic report structure (header, body, and footer)
- Extending the basic structure into subsections for complex reports
- Essentials for creating better reports
- Various industry-standard reporting patterns
- How different reporting patterns are used in the real world

Reporting Dynamics

Client-side reporting means producing reports on a local (client) computer rather than on a central server. When Microsoft (MS) introduced MS Access 1.0 in the early 1990s, producing reports simply and with minimal effort became possible. As we moved through the decade, the era of web reporting raised new challenges as the Web became ever more popular for delivering information. Although it was easy enough to produce reports and display them on web pages, other issues, such as printing attractive reports, were not easy to resolve. Visual Studio now has built-in support for creating professional-quality reports for various clients.

Before VS 2005, Microsoft provided various choices for client-side reporting. The most common is the Crystal Reports (CR) add-in from Business Objects. If you come from the world of Visual Basic 6, then you'll also know Data Report Designer. With the introduction of Reporting

Services (RS) at the client-side with VS 2005, Microsoft has given developers a serious tool for producing reports. Don't worry if you are already skilled with CR—in Chapter 12, we'll look at CR and RS in action side-by-side to help you transition to RS.

Efforts by Microsoft to provide an alternative to CR prior to VS 2005 were, at best, filling the blanks and were no serious challenge to the capabilities of CR. The .NET platform with Visual Studio came as a real opportunity to provide a simple architecture to produce client-side reports with access to all sorts of data.

This book teaches you how to use VS 2005 and Visual Studio 2008 RS for all your reporting needs through hands-on practice with the software. Before we start using RS, though, let's look at reporting architectures and environments in general.

Client-Side Architecture

The architecture of client-side reporting isn't rocket science. Everything revolves around your client application, as shown in Figure 1-1.

Your client application gathers data from your favorite data source and processes the report definition to produce a report. The ReportViewer presents the result to users. The interesting idea here is that all steps needed to produce a report are an integral part of the client. The steps needed to create the report definition and to collect data are identical for all supported clients.

VS 2005 and Visual Studio 2008 will create the report definition (as shown in Figure 1-1). ADO.NET is commonly used to collect data from the source. The last step is to bind the data with report definition using ReportViewer. The client application produces the report, with report definition technology embedded in the application or loaded from a disk.

Client-Side Architecture Characteristics

Let's briefly explore some of the characteristics of client-side reporting architectures and the terminology used to describe them.

Data Source

We know how important data is for creating reports. At times, getting hold of the data is easy, but at other times, it is a challenge. Let's keep it simple and say that data that ADO.NET can collect and load into a DataSet is a valid source.

But don't worry with terms such as ADO.NET and DataSet now. Just keep in mind that ADO.NET is a data interface provided with VS, and DataSet is a placeholder for the data that is used for reporting. ADO.NET is versatile enough to connect to a variety of data sources ranging from sophisticated relational databases to file-based text data. You can learn more about ADO.NET and DataSet in Chapter 3.

■ **Note** It is a common misconception that VS Reporting Services is a SQL Server extension and can only report on SQL Server data. Both server-side and client-side RS report from a wide variety of data sources, including flat files and databases such as Oracle and MySQL.

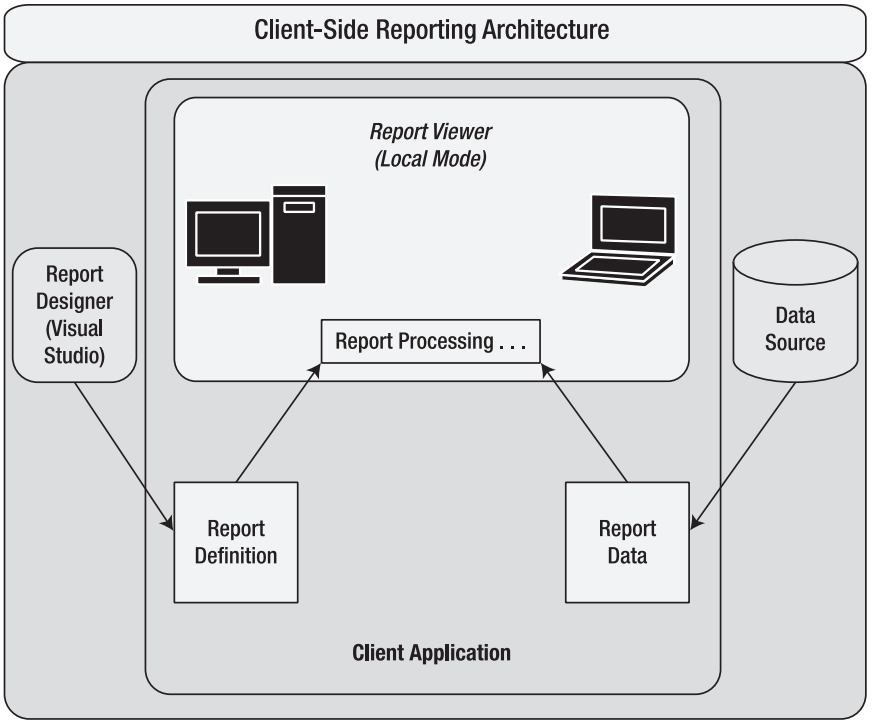


Figure 1-1. *Client-side reporting architecture*

Report Definition Language (RDL)

Report Definition Language (RDL) is an XML-based file. The report designer automatically produces the file, which has an `.rdlc` extension (the "C" stands for "client-side").

■ **Note** You don't need to know XML or RDL to create reports. RS handles everything for you transparently.

All objects added to a report in the report designer are part of an RDLC file. Since XML is an open standard used by independent software vendors and custom solution developers, it is easy for them to create interface tools to interact with RS.

RDL also makes it easy for report designers and developers to open the report definition in a text editor to make changes outside the report designer. Here's a small snippet showing how a text box item is stored inside an RDLC file:

```
<Textbox Name="ProductNumber">
  <rd:DefaultName>ProductNumber</rd:DefaultName>
  <ZIndex>3</ZIndex>
  <Style>
    <PaddingLeft>2pt</PaddingLeft>
```

```
        <PaddingBottom>2pt</PaddingBottom>
        <PaddingRight>2pt</PaddingRight>
        <PaddingTop>2pt</PaddingTop>
      </Style>
      <CanGrow>true</CanGrow>
      <Value>=Fields!ProductNumber.Value</Value>
    </Textbox>
```

So, you might be wondering, what is the difference between RDL and RDLC? Well, the main one is that RLDC files don't store any data query or data connation information. RDLC files can easily be hosted as RDL files on a report server with little effort.

Because RDL is XML based, it is easy for third-party vendors to come up with RS extensions. For example, how about printing a barcode? Well, that's easy enough with a component from Neodynamic. Watch out for a cool report based on a barcode component in Chapter 13. Or, you can get a trial version of Neodynamic's Barcode Professional software here: `http://www.neodynamic.com/Products/BCRS/BarcodeRS.aspx?tabid=78&prodid=7`.

If you think you would like to come up with your own RS extension, you can download the copy of the RDL specification at `http://www.microsoft.com/sql/technologies/reporting/rdlspec.mspx`.

Report Parameters

Report parameters are the key means of communication among report definitions and report requests. Most reports are designed to provide dynamic information based on criteria provided at run time. For example, if you had 50 employees in your organization, would you develop 50 different pay slip reports? No, you'd dynamically pass the Employee ID as a parameter to produce a report for each employee.

Previewing and Exporting

In VS, the most common way to display a report to users is the Preview feature. The Preview feature not only shows how a report looks but also lets users print reports. Besides preview and print capabilities, client-side reporting also offers the ability to export reports in Excel and PDF formats. You don't need separate software to render a report in PDF format.

Security

Client applications manage the security. Custom code used in reports must have permission to access the file system or network. For example, if your report is trying to read data from an XML file over a network, the application account must have proper rights to access that network.

■ **Note** RS has the same user interface in both VS and SQL Server. Client-side reports deploy as part of client applications; there's no need for a SQL Server license to use RS in VS.

Server-Side Reporting Architecture

This book is about client-side reporting, so why look at server-side reporting architecture? It is good to know both sides of the technology. Imagine if you need to port some of your client-side reports to the server-side? Also, it is interesting to make a contrast here, to better understand the client-side architecture by looking at the server-side counterpart. Figure 1-2 shows the server-side reporting architecture.

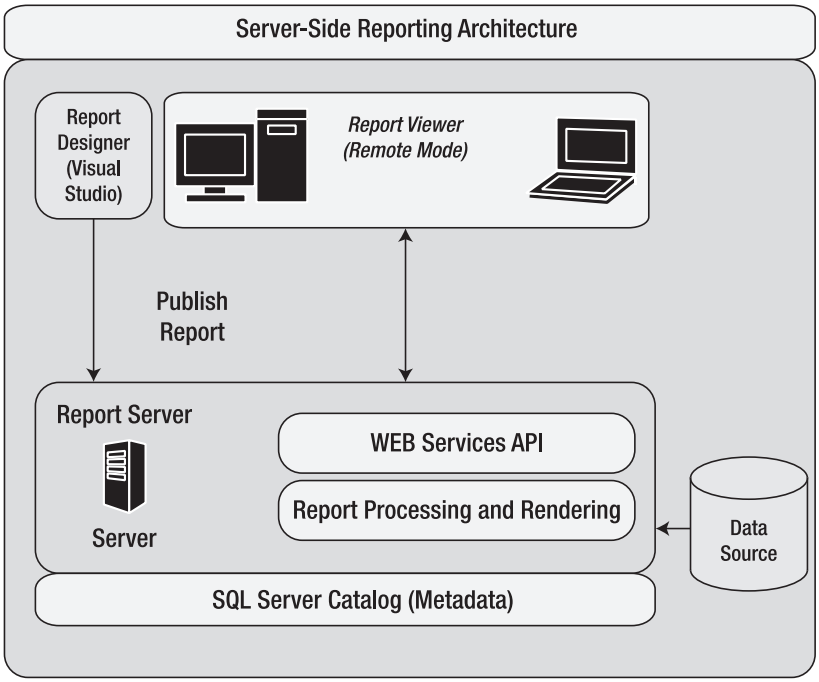


Figure 1-2. *Server-side reporting architecture*

There are a few differences between how the report gets produced by client-side and server-side RS architecture. Let's start with data. We can only report on data that is accessible through the reporting server, as opposed to getting everything that ADO.NET can bring for client-side reporting. The report server publishes the report on the server side.

ReportViewer manages user input to parameters on the server side; for a client-side report, you'll need custom user interface (UI) code. All export formats are supported by RS on the server side, including Excel, PDF, and MHTML. Client-side RS only supports Excel and PDF formats.

Users

People look at reports in a different ways, so a single report can have a different impact on various people. For example, a student progress report can say a lot to parents about a child's performance. On the other hand, the same report can help the school principal check the performance of teachers.

In almost all organizations, reporting perspectives vary in a hierarchical way. Executives mostly focus on summaries that help them decide about finances and future directions. Line-level managers and workers care about various details.

Informed decision making is important at all levels within an organization, and easy access to accurate information is also essential. It is of the utmost importance that developers understand both their users and their users' information needs, so they can design and produce effective reports.

It's not possible to describe every potential report user. However, a few generic classifications of the business-reporting audience are useful.

Customers

Developing reports to satisfy the needs of customers is a huge challenge. Customers come in various classifications, from one-time buyers to those who appear in a company's top ten list based on their sales volume. Trust me; both are easy to manage with a client-side reporting solution. You can easily develop a Sales Invoice report for one-time customers and an aged receivables report to track unpaid invoices for your regular customers.

You've probably heard the term business-to-consumer (B2C). So, how can client-side reporting communicate valuable information from business to consumers? Well, a common example is an Order Status report hosted on a corporate ASP.NET site to let customers track the shipping status of their orders.

Client-side reporting technology can help you develop reports to provide both current and historical information on demand to the customer. You can deliver reports through a web interface, e-mail, or mobile connectivity to improve the customer experience.

Vendors

Consumers deal with vendors primarily for two reasons: to buy a product or a service. This is often called business-to-business (B2B) communication. Client-side reporting supports both types of activities.

Vendors interact with businesses in different ways, according to the need. For example, you could develop a Stock Reorder report or a Fast-Moving Items report to share your stock consumption information for better handling of stock procurement.

The trend to automate business transactions is a common practice. For example, an automated Purchase Orders report can feed into a vendor's order system to speed up the purchasing process. Although business transactions carry lot of challenges, a properly done report can improve the interactivity among business and vendor partners.

Executives and Managers

Executives rely primarily on summary data and find graphical representations powerfully persuasive. For example, at the end of each fiscal year, executives may study a series of reports such as yearly sales analyses, budgeted versus actual expenses reports, and balance sheets to see how a business performed during the year.

Cool graphs and charts make the maze of numbers inherent in reports much more understandable. An intranet-based portal is a popular choice to deliver reports to executives, and it's supported by client-side reporting technology.

Line Managers

Line managers look for reports that are more detailed in nature. Reports that line mangers might need, like raw materials status or worker shift allocation reports, are easily handled on the client side.

Everyone Else

No matter what you do, some kind of information is useful. With VS RS, anyone can produce both detailed and summary reports from an extraordinarily wide set of data sources to meet virtually any information need.

■ **Note** Report names, such as the Stock Reorder report, are mentioned in the previous section to tell you about what's available for use in various situations. You'll get a chance to practice the reports when we look at practical examples, starting in Chapter 4.

What Applications Are Supported?

In this book, we'll discuss four different kinds of client applications and develop various real-world examples with them. The client applications are based on the following technologies:

- Windows Forms
- Web Forms
- Web services
- Windows services

■ **Note** Brief tutorials on how to use these client applications appear as they come up in this book, so don't worry if you're not an expert now.

Windows Forms

Windows Forms has improved a lot over the years. Better security features and ease of deployment have made this client a perfect host for client-side reporting solutions. Using VS RS, you'll develop a Windows Forms project and use the ReportViewer control to preview a report.

Which report is the best choice to host with the Windows Forms client? Well, it depends on what the report content is and who needs to see it. In Chapter 4, I'll guide you step-by-step through developing the most common reports in real-world desktop applications. For example, we'll create a Trial Balance report, which is a common feature of accounting applications.

Web Forms

ASP.NET Web Forms is a sound platform for reporting on both intranets and the Internet, and it's the easiest way to share reports with large numbers of users. In Chapter 5, you'll create an ASP.NET web site project with VS RS and host the report using the ReportViewer control on the page. With VS RS, you don't have to worry about application deployment to a client machine; all users need is browsing software, and they are ready to go.

The reach of the Web Forms application is much greater than Windows Forms applications. After going through the Web Forms tutorial in Chapter 5, I'll show you how some of the most common reports are developed using client-side reporting. For example, you'll create an order tracking report, and I'll show you some cool tricks on how to plot graphs in it.

Going wireless is getting increasingly popular among businesses. This trend has brought new challenges to the developer community as developers look for on-demand information-delivery solutions. Reports done with web pages can be easily shared with various smart devices, and therefore, give true meaning to concept of accessing information anywhere and any time.

Web Services

A web service is a technology designed to enable information sharing among the different systems over any given network, such as the Internet and an intranet. Web service can also help to eliminate the duplication of business functions that are commonly shared by more than one application.

If you are new to web service development, I'd strongly suggest you to go through the tutorial in Chapter 6. In that chapter, you'll also see how a report can be generated on demand by calling a web service method and sent across to the consumer client.

Windows Services

Windows services are a neat way to automate tasks that need to be performed on a predefined interval. Using VS RS, you can create a Windows service and automate it. Since this is a service, users will not be able to see the report preview; instead, the report will be exported to either Excel or PDF format.

Never developed a Windows service before? Not to worry, your friendly tutorial is standing by in Chapter 8. There are many opportunities to automate and deliver client-side reports using Windows services. I'll show you, among other things, step-by-step how to create a report called New Complaints report and send it to a customer service manager's e-mail box on a selected time interval.

The Report Creation Process

What if I said the report creation process is as simple as 1, 2, 3? Well, it may not be that easy, but I'm confident that you'll love the straightforward approach of creating reports in VS. Figure 1-3 shows the process, which is explained in more detail in the following sections.

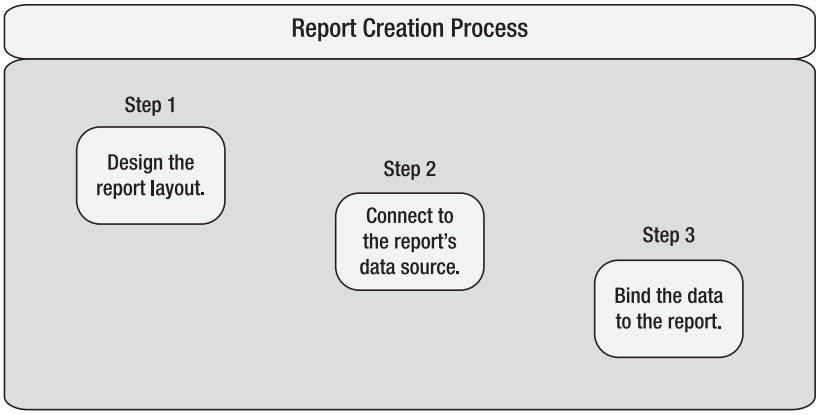


Figure 1-3. *The report creation process*

Designing the Report Layout

The best report layout is the one that reflects the nature of the data. Summary data is a good candidate for charts and graphs. Detail data often needs a list or drill-down approach. Industry-standard best practices are common for report layouts; ever wonder why all bank account statements look so similar?

Connecting to the Data Source

At times, data access is as simple as connecting to a local SQL Server or Access database. It becomes more challenging if we need access to a business application object or proprietary data source. How many different data sources can we use for a report? Typically one; however, for complex reports there can be more, and they can be of different types. For example, you can use data from both a SQL Server and another database in the same report.

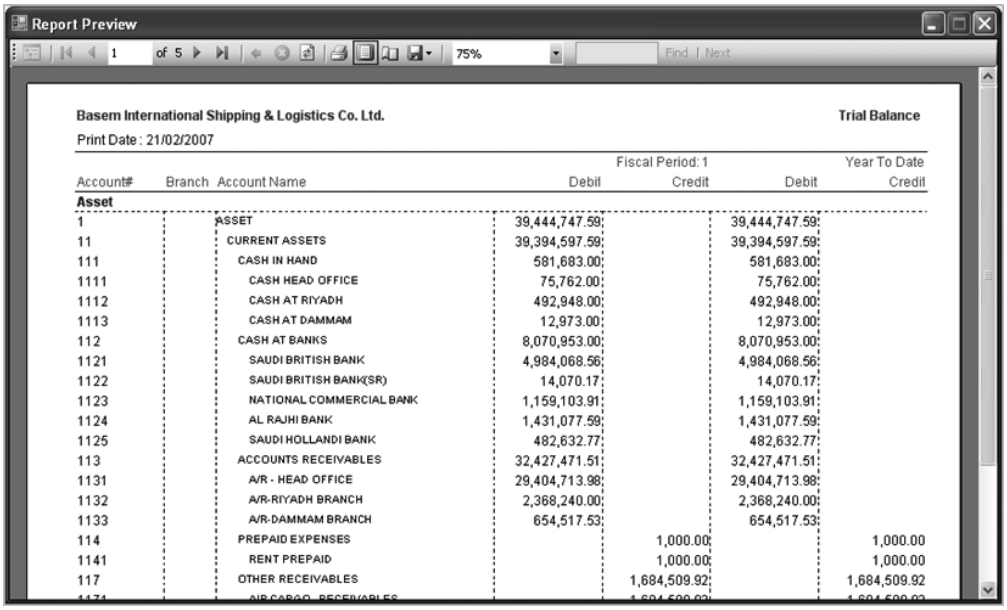
■ **Note** Data quality is another important issue. As the saying goes, "garbage in, garbage out." It's wise to always find out if we have any garbage to deal with and make sure it doesn't become a showstopper later.

Binding the Data to the Report

After taking care of the data source and designing your layout, all you need is to issue a few magical statements to pump life into your report. Binding data to the report programmatically is the same for every supported client, as mentioned earlier, and you can use any scripting language supported by VS to do it. We'll use C# for all the examples in this book.

A Real-World Report in Action!

I've made several references in this chapter to real-world report examples from the business world, and you'll practice many of these reports throughout the book. For now, let me show you what a real-world report looks like. Please see Figure 1-4 for a sample Trial Balance report that I've done for one of my clients (although the report is real, it uses mock data).



Basem International Shipping & Logistics Co. Ltd. **Trial Balance**

Print Date : 21/02/2007

Account#	Branch	Account Name	Fiscal Period: 1 Debit	Fiscal Period: 1 Credit	Year To Date Debit	Year To Date Credit
Asset						
1		ASSET	39,444,747.59		39,444,747.59	
11		CURRENT ASSETS	39,394,597.59		39,394,597.59	
111		CASH IN HAND	581,683.00		581,683.00	
1111		CASH HEAD OFFICE	75,762.00		75,762.00	
1112		CASH AT RIYADH	492,948.00		492,948.00	
1113		CASH AT DAMMAM	12,973.00		12,973.00	
112		CASH AT BANKS	8,070,953.00		8,070,953.00	
1121		SAUDI BRITISH BANK	4,984,068.56		4,984,068.56	
1122		SAUDI BRITISH BANK(SR)	14,070.17		14,070.17	
1123		NATIONAL COMMERCIAL BANK	1,159,103.91		1,159,103.91	
1124		AL RAJHI BANK	1,431,077.59		1,431,077.59	
1125		SAUDI HOLLANDI BANK	482,632.77		482,632.77	
113		ACCOUNTS RECEIVABLES	32,427,471.51		32,427,471.51	
1131		A/R - HEAD OFFICE	29,404,713.98		29,404,713.98	
1132		A/R-RIYADH BRANCH	2,368,240.00		2,368,240.00	
1133		A/R-DAMMAM BRANCH	654,517.53		654,517.53	
114		PREPAID EXPENSES		1,000.00		1,000.00
1141		RENT PREPAID		1,000.00		1,000.00
117		OTHER RECEIVABLES		1,684,509.92		1,684,509.92

Figure 1-4. *A real-world report sample (Trial Balance)*

Let me also introduce you to one more cool feature of RS here. Do you think RS can develop reports only in English? How about reporting in other languages such as French or German? Well, you'll be glad to know that RS comes with strong support for developing international reports. Yes, that includes the languages that are read from right to left, such as Hebrew or Arabic. Figure 1-5 shows the Trial Balance report again. However, this time the report is in Arabic.

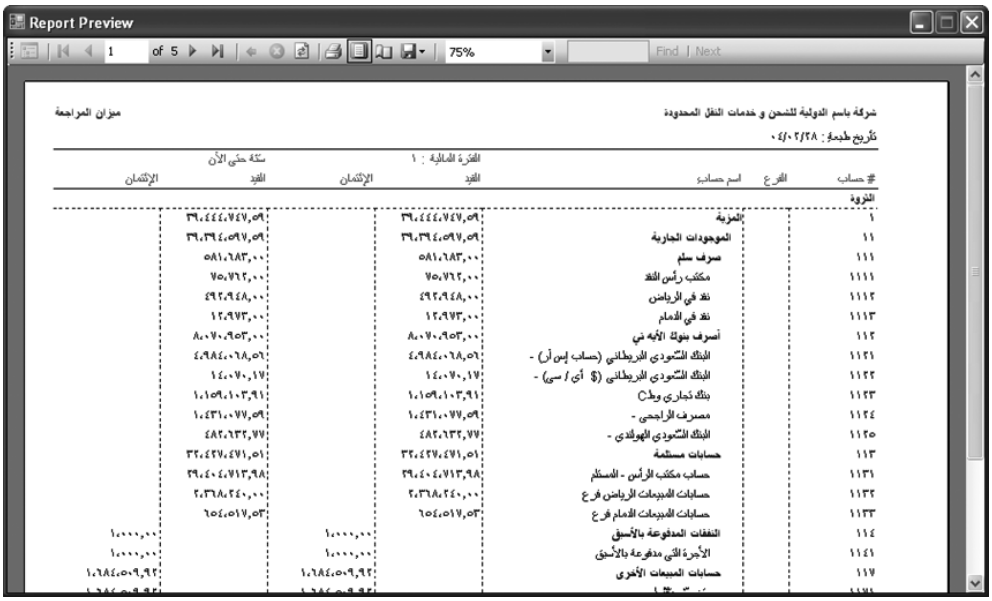


Figure 1-5. *A real-world report sample (Trial Balance) in Arabic*

A Hands-On Approach

When I started working on this book, I asked myself how I could make this book a guided tour for learning client-side reporting. Sure, I could've just asked you to click here, drag this, and change that to create a report. However, my goal is to teach you not only how to create a report but how to develop practical, professional, real-world reports. After much thought, I came up with the following approach, and I hope that it will be comfortable for you:

- *Challenge*: Define the reporting project.
- *Process*: Apply the three steps required in developing a report.
- *Practice*: Enhance your understanding with exercises; solutions are provided (but no peeking).

Challenge: Defining the Reporting Project

The most difficult decision for me was to decide which real-world report case study to present as a hands-on project throughout this book. By using a chart of accounts report, I would make accounting folks happy. However, what about folks who don't understand the language of debits and credits? What if you're an expert in human resources systems and hope to walk through a sample report in that field?

Covering each practical report out there in the world of business is not possible in this book. Therefore, I decided to pick the most common report type and one that has an easy-to-understand business case—the Chart of Accounts project. Even if you don't know accounting, I promise after going through the Chart of Accounts project, you'll sure learn a secret or two to brag about it in front of your accountant buddies.

I'll start with a general discussion of the business case for the report and show you a screen shot of what the finished report should look like. Your goal is to understand the underlying business case of the report and start the step-by-step guided tour of report development.

The business case will look like something you might have seen in a certification examination like the Microsoft Certified Professional (MCP) exam, for example, "You're a software developer with ABCD Pharmaceuticals, Inc. The company has ten profit centers in the United States and five in Canada. You need to develop a report that will show combined sales figures from all 15 profit centers on a quarterly basis."

Process: Applying the Three Steps in Developing a Report

The process of developing the report will be a bit challenging at first. However, as you move on from one project to another, you'll find yourself growing increasingly at ease. If you are at the beginner level, make sure you understand the business case thoroughly. Business cases can dictate a lot about what the report layout should look like and which report pattern is best to use. I'll show you how to develop the reporting project with the help of the three-step process shown in Figure 1-3.

I'd strongly advise you to start with the practical examples in Chapter 4, if you consider yourself at the beginner level; I've placed extra emphasis on explaining every detail to help you understand each step in the report creation process. While going through Chapter 4 and beyond, you'll notice that the complexity of examples will grow as you move through each chapter's content. All reporting projects discussed in this book will have many similarities among them. However, you'll also notice differences depending on the business case involved.

The process has the following three steps:

1. Create a dataset.
2. Design the report layout.
3. Write the C# code to gather the data and bind it to reporting engine.

Creating a Dataset

Can we create a report if we don't know what the content is? Obviously not—it is the data that helps in deciding about the report pattern. The first step is to create a typed dataset with one or more data tables inside.

Typically, the data table inside the dataset should use the same structure as the data source. However, the data table structure inside the dataset can have additional columns that are different than the data source. Such columns are manipulated and filled with data programmatically at run time before binding the report to the host. For example, a dataset can have the columns `FirstName`, `LastName`, and `FullName`, where `FullName` is an extra column that is filled in `DataSet` and not supplied by the data source.

I'll identify the data source to you from the `RealWorld` database, which has a variety of different business entities. You'll get a `RealWorld` database with the source code for this book, and I'll show you step-by-step how to map the data source to a data table inside the dataset. I've set aside Chapter 3 to discuss entities of the data model used with each report project; I'd advise you to use that chapter as a reference to the `RealWorld` data model.

■ **Note** The dataset is a memory-resident representation of data that provides a consistent relational programming model regardless of the data source. The dataset page of MSDN can help you understand datasets in more detail: `http://msdn2.microsoft.com/en-us/system.data.dataset.aspx`.

Design the Report Layout

After taking care of the dataset, the next step is to make use of the report design surface. The nature of the data content is the first hint to deciding what report pattern to use. For example, if your data involves grouped items, the best choice is a tabular report pattern, while a graphical chart report is the best choice to present summary data.

You'll design the report step-by-step starting with the report structure, next picking relevant report items, and finally organizing them on the report design surface. You'll also learn how to give a professional look and feel to the report and see ways to beautify, such as setting its colors, shape, size, and fonts.

Writing the C# code

C# coding is the important final step to populate a dataset from one or more data sources. User input is sent across to the report using parameters. You'll get step-by-step guidance to get the data, work with parameters, and finally, bind the report to the client using the ReportViewer control.

■ **Note** Although we will be using C# to interface with the ReportViewer control, RDL files only support VB.NET for scripting within the report.

Practice: Enhancing Your Understanding with Exercises

In Chapter 4, you'll see some exercise to practice and apply the basic report development knowledge you just learned. I'm preparing you for more challenging reports in Chapter 5 onward by offering you these exercises. Although solutions are provided, I'm sure there will be *no peeking*! First, try it out yourself.

Report Structure

Every report design assignment has its own challenges. Like a painter who changes the landscape and colors in a painting to suit the subject, developers also change report structure to best suit reported data. The most basic report structure includes three sections, as shown in Figure 1-6:

- *Header*: Ideal place for items such as the report title and date
- *Body*: The details of the report, whether summary or detailed business transactions
- *Footer*: Ideal place for items such as page numbers and copyright information

---

■ **Note** The body is also commonly called the details section.

Most real-world reports are developed using these three basic sections. However, some business cases bring additional complexity to reports. The most common adjustment is to add subsections to the default layout. In such cases, we will need to adjust the structure. Before we look into the details for each section of the report, let's examine the overall structure in Figure 1-6.

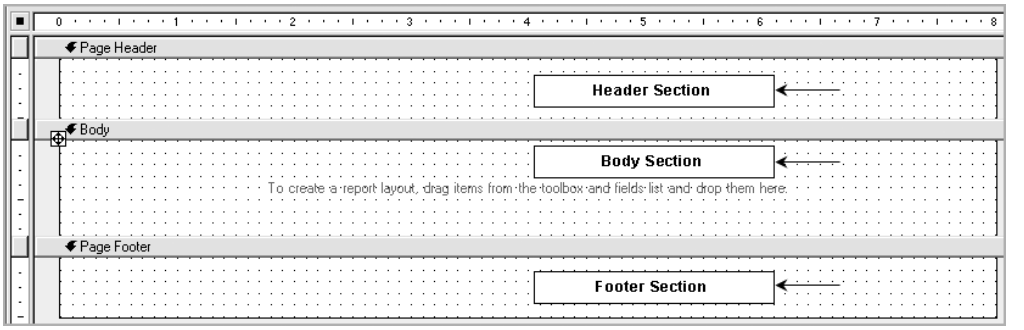


Figure 1-6. *The basic structure of a report*

VS RS offers this basic structure for you to start the design process. Notice in Figure 1-6 that the report design surface looks like a paper page; a page is like a drawing board or canvas. All you have to do is start putting your design skills to work and convert this blank canvas to a report with relevant information.

■ **Note** When we add a report to a project, only the body section is automatically added; you'll need to manually add the header and footer later.

Header

The header section gives recognition to the report. How? Simple—the majority of the report's titles or names find a place in the header section.

So, what else goes into the header section apart from the report title? The following items qualify as good candidates to appear in the header section:

- Name of the organization
- Organization's logo
- Report date and time
- Drawing elements, such as a line or rectangle
- The supplied report parameter values (possibly)
- Page numbers (more commonly a footer section candidate though)

Please see Figure 1-7 for a simple illustration of what typically goes into the header section and how it helps the end user to understand the origin and subject of the report.

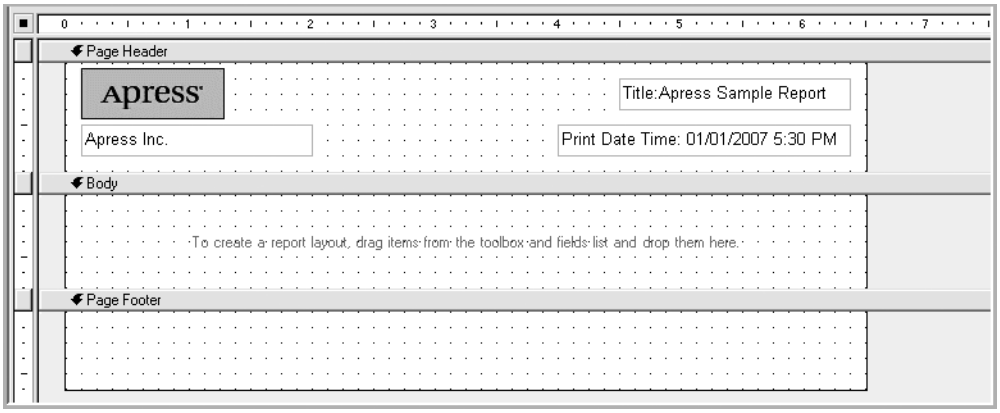


Figure 1-7. *Typical content for the header section*

Body

The body section is, by far, the most versatile of all three sections and is my favorite on the report design surface. I'd even go so far as to say that anything and everything can be placed here—from numbers to text to graphics, all information is welcome.

Out of the many different possibilities, the following most often appear as content in the body section:

- Summary totals
- Transaction details
- Grouping and sorting of data

- Multicolumn output
- Charts and graphs
- Links to other reports
- A specialty report, such as a promotional letter

Please see Figure 1-8 for a simple illustration of the body section; this one shows how the detail section uses payout detail transaction information to show bonus information.

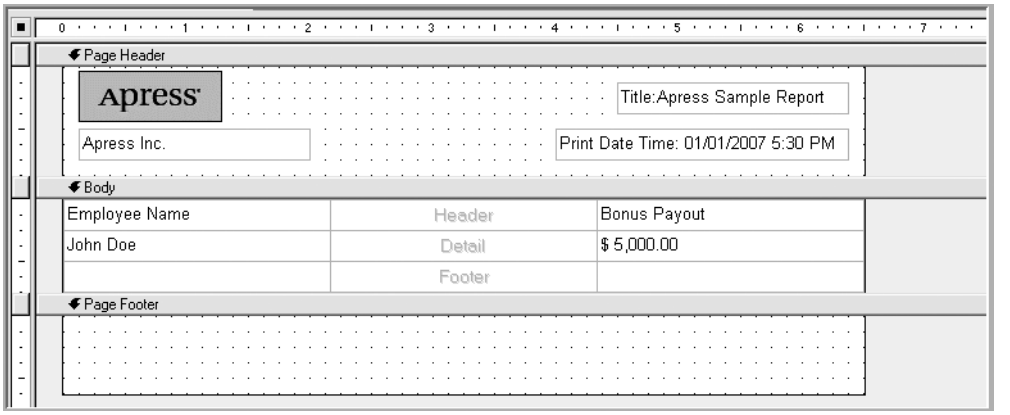


Figure 1-8. *Typical content for a body section*

I'm sure you've got a question for me now, "What are those additional header, detail, and footer labels doing inside the body section?" I know you're all excited to develop some cool reports with client-side reporting, so starting in Chapter 2, I'll show you each element of the Report Designer and Print Preview controls. I've used a table in the body section to show how a detail report can make use of this section to list column names, data details, and the total of all listed transactions.

Footer

The third and the final section in the report structure is the footer section. I've seen developers use the footer to show running page numbers in the report as a common practice. Developers enjoy the freedom VS offers in placing content in the header to footer according to user need. Other miscellaneous information, like copyright text, also makes it into this section. At times, even the physical file name of the report is displayed here for reference.

Please see Figure 1-9 for a simple implementation of the footer section, showing page numbers and a line separator.

Take a look at Figures 1-6 to 1-9 again. What you see is a gradual transition from the blank report design surface to a complete report. You, as a developer, will go through countless adventures like this one to produce stunning reports that I'm sure you'll be proud of.

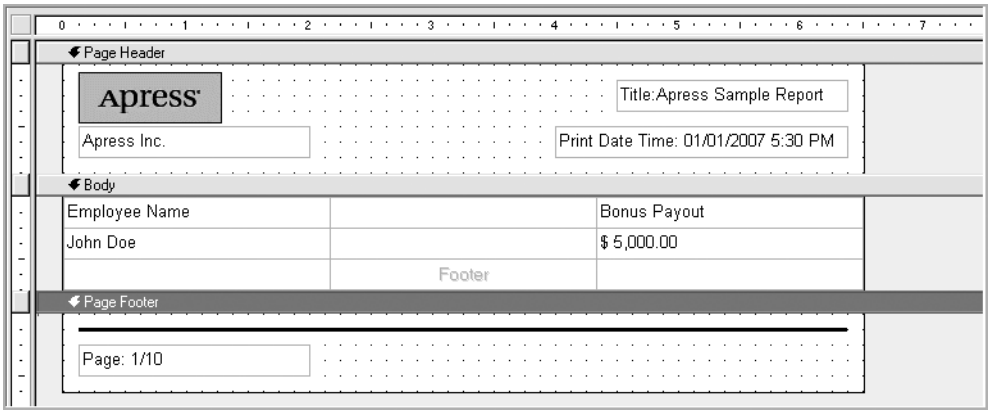


Figure 1-9. *Typical content for the footer section*

Creating Better Reports

How do you create those stunning reports? The following practices are widely used by report developers to come up with better report designs:

- Sketch the report layout on paper first.
- Decide on the page layout, paper size, and orientation (portrait or landscape).
- Determine the space allocations and formats for all data.

Once you've done this, you're ready to create the report with VS RS.

Report Patterns

The report design surface in RS is not like any other report designing tool you've used. In appearance, it's similar to Access or Active Report. However, you'll feel the difference once you start to develop reports. Let's imagine you've got a canvas; as you would mix and match different colors and brush strokes to fill the canvas, do the same with report items. Let your creative imagination run wild.

Naturally, with all this freedom on your side, you may want some patterns to focus your efforts. How many different reporting patterns can you use to design your reports? The most commonly used patterns in the real world follow:

- Form/page
- Tabular details
- Multicolumn
- Grouped data
- Matrix/pivot table

- Chart/graph
- Drill-through
- As you like it

The Form/Page Pattern

This pattern is largely popular in real-world report business cases to report data that is a single piece of information. The information could be anything, for example, a single record from a table or some static text. If you're developing a sales invoice or receipt voucher, you're already making use of this pattern.

Please see Figure 1-10 for a sample sales invoice report.

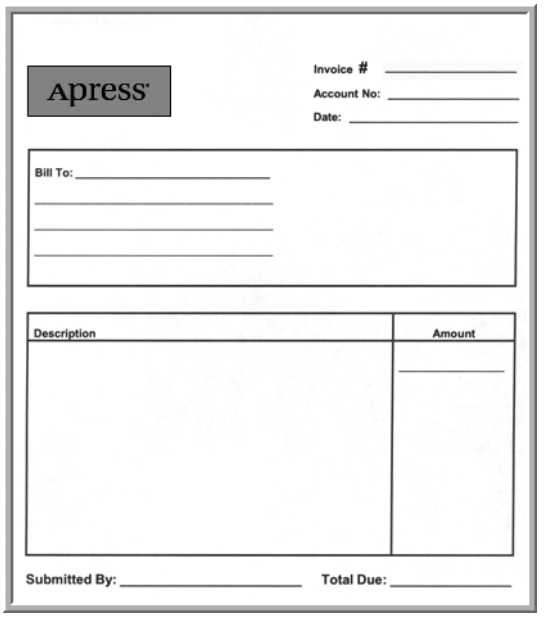


Figure 1-10. *An example of the form/page pattern*

The Tabular Details Pattern

The tabular pattern is the choice of many developers when it comes to reports with multiple lines of transaction details. Why is this pattern so popular? The content of this report presents detailed business transactions in a format like a spreadsheet, with rows and columns. In addition to transaction details, summary totals are also part of the report. Please see Figure 1-11 for an example of the Product List report, which displays product information and the total stock value.

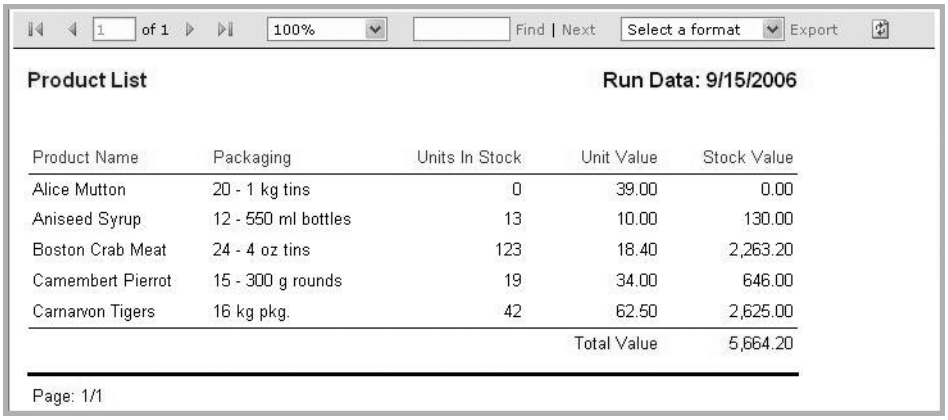


Figure 1-11. *An example report using the tabular details pattern*

The Multicolumn Pattern

Once, when I was developing a sales ordering system for one of my clients, I had to produce a report called Summary Price List. What made this report interesting to me was the user need: the user wanted to save large numbers of pages and still have a hard copy of the price list with only the Catalog ID, Cost Price, and Discount Factor columns.

The multicolumn pattern came to my rescue; I created a three-column report with a landscape page orientation and reduced the overall page count by ten percent, compared with the report with a single-column portrait format.

■ **Note** The multicolumn pattern is also a popular pattern to develop labels and contact cards for customers.

The Grouped Data Pattern

This pattern is similar to the tabular pattern. However, what makes it different is the ability to report data in a grouped and sorted manner. Let's look at this pattern with the help of an example. Please see Figure 1-12; this is typical tabular report with a list of orders as details. If you notice, all orders are listed as part of a group, and they appear under the heading of the customer name. Grouping information based on certain business criteria helps the user to access the information much easier.

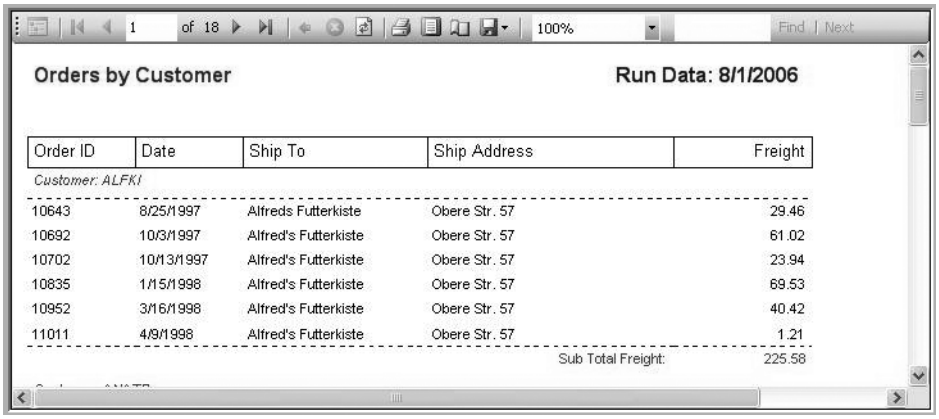


Figure 1-12. *An example of the grouped data pattern*

■ **Note** The drill-down report pattern is similar to the grouped pattern, but in addition, the user can collapse or expand the group information while previewing the report.

The Matrix/Pivot Pattern

The matrix/pivot report pattern, also referred as cross-tab, is widely used to report summary data by rolling up rows of details into columns. Being a developer, if you've ever tried to create a cross-tab report, you know how much of a serious effort it is. Some developers make use of SQL query or some write custom code to mimic the matrix pattern.

Good news! RS has made your life easy. All you have to do is drop the Matrix Report item onto the design surface and set the data source. You'll be surprised to see how easily your data is changed into a pivot table. Please check Figure 1-13 for an example of data with rows and columns converted into pivot output.

Order ID	Fruit Type	Quantity
1001	Apple	10
1002	Mango	2
1001	Apple	11
1003	Banana	55
1004	Mango	34
1005	Banana	11
1006	Apple	2
1006	Mango	3
1007	Mango	4
1001	Apple	6
1002	Banana	6
1005	Banana	10

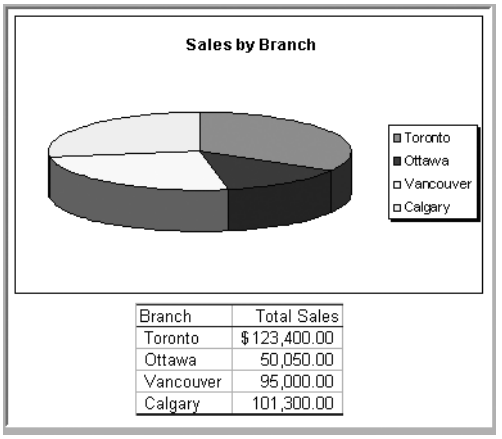
Sum of Quantity	Fruit Type			
Order ID	Apple	Banana	Mango	Grand Total
1001	27			27
1002		6	2	8
1003		55		55
1004			34	34
1005		21		21
1006	2		3	5
1007			4	4
Grand Total	29	82	43	154

Figure 1-13. *A matrix/pivot pattern example*

The Chart/Graph Pattern

We often hear that a picture is worth a thousand words. Yes, at times instead of putting plain numbers on a report, it is better to present them using graphics. Executive and management users prefer this report pattern most often. Some of the most common chart types used in reporting are bar, column, pie, line, and area.

Please see Figure 1-14 for example of the chart/graph pattern.



Branch	Total Sales
Toronto	$123,400.00
Ottawa	50,050.00
Vancouver	95,000.00
Calgary	101,300.00

Figure 1-14. *An example using the chart/graph pattern*

The Drill-Through Pattern

The drill-through report pattern is unique in that any report can be a drill-through report. You might be wondering how. Well, it's possible because of something called a *link*; an embedded link allows the user to launch another report by clicking it. Any report that includes links is a drill-through report.

A typical example would be a receivables report, in which the user can click a highlighted link, such as an invoice ID, to launch the corresponding invoice report. You're running a report from within another report.

The "As You Like It" Pattern

Hmm, I can imagine that grin on your face. Like me, you will probably spend much of your time experimenting with this pattern. I've said a couple of times that the report design surface is like a canvas, and that's especially true of this pattern. Start from any direction; draw any control; mix and match any number of patterns; in short, have a fun time painting your ultimate creation.

How about developing a personal dashboard with an RSS feed from your favorite news channel? Yes, this all can be done. Look for your chance to practice this in Chapter 13.

The example figures in this section are just for you to get the concepts. The real work of designing using different patterns will be covered starting in Chapter 4.

Summary

In this chapter, we discussed the architecture of client-side reporting. We also looked at the typical users of client-side reports, the client applications we'll work with in this book, and the three steps for creating reports. I also touched on the report structure, real-world reporting patterns, and best practices for creating reports.

In the next chapter, we'll look at the client-side components offered by VS RS to produce stunning, powerful, professional reports.