Crystal Reports™ 8.5 Developer's Guide

Seagate Software IMG Holdings, Inc.
915 Disc Drive
Scotts Valley
California, USA 95066

Documentation Issue 1. Febuary 2001.

Contents

What's New for Developers 1

With the release of Crystal Reports version 8.5 developers edition, Crystal Reports continues to improve the flexibility and power of the Report Designer Component, along with it's web reporting capabilities. New components and features allow you to enhance your users reporting experience and increase the availability of Crystal Reports.

This chapter introduces the new features and enhancements, which fall into three key cateories.

- Report Designer Component (RDC)
- Crystal Report Viewer Control
- Web Reporting.

Web Reporting

To better meet your web reporting needs, Crystal Reports now integrates fully with Crystal Enterprise, a web-based report management tool that works within your company's existing web infrastructure.

If you purchased the Professional or Developer edition of Crystal Reports, then you also received the Crystal Enterprise Standard CD. This additional CD is included in the box with Crystal Reports, along with five free concurrent access licenses of Crystal Enterprise Standard.

- Improved interactivity with parameters
- Customizable DHTML Report Viewer
- Display web reports independent of server location
- Extend reporting abilities with additional applications

Increased value of the Crystal Enterprise Web Component Server

Use the newly re-designed and enhanced Web Component Server (WCS) from Crystal Enterprise to power your web-reporting solution and to replace the older WCS from Crystal Reports 8. Crystal Enterprise Standard provides significant improvements and enhancements over the version 8 WCS.

Run the Web Component Server on any machine

Install the Crystal Enterprise Standard CD on whichever machine you want to coordinate your web reports. With Crystal Enterprise Standard, you no longer need to install and run Crystal Reports and the WCS on your web server.

If you need a reporting solution with greater scalability, use Crystal Enterprise Professional, which can be installed on as many machines as are necessary (whereas Crystal Enterprise Standard is limited to installation on a single machine). Further, with Crystal Enterprise Professional, you can specify the server pieces you want running on each machine.

Scale your reporting framework to meet increasing demands

Crystal Enterprise provides you with a scalable, web-based solution for managing the access and delivery of hundreds or thousands of mission-critical Crystal reports to every decision-maker—across the enterprise and beyond.

As your company's web reporting needs increase, you can upgrade seamlessly to Crystal Enterprise Professional without having to reinstall; you can also add more licenses and components so that Crystal Enterprise grows right along with you.

Schedule reports to maintain up-to-date information

Schedule important reports to run on a regular basis so that everyone has access to the most current information about your enterprise.

Publish reports to the Web in seconds

With just a few easy steps, you can publish your reports to the Web for viewing by all with the Report Publishing Wizard.

Manage folders for sharing reports

Share reports across the enterprise or across the Web by publishing them to the default Guest favorites folder on the Automated Process Scheduler (APS). Users can also publish to other folders, as determined by the Administrator.

Administer easily from one central console

By consolidating administrative tasks, the Crystal Management Console (CMC) makes administering your web reporting solution quick and easy—regardless of the size of your enterprise.

Use the CMC to gain immediate control over what can or cannot be seen, run, or managed by everybody. You define all users' viewing and managing privileges for particular reports.

Integrate Crystal Reports performance with existing IT investments

Trust your Crystal web reporting solution to work to its full potential with your existing web server investment.

Crystal Enterprise Standard has improved web server support, including compatibility with Domino web servers through DSAPI interface, and Apache servers through a DSO (Dynamic Shared Object) module on Solaris and Linux. Crystal Enterprise Standard also supports CGI on Solaris and Linux.

Use scripting on the Crystal Enterprise WCS

The Crystal Enterprise Web Component Server now provides complete support for scripting.

Complete XML support for the Web

Crystal Reports now fully supports the Extensible Markup Language (XML) adopted by the World Wide Web Consortium (W3C) for delivering content over the Internet.

XML is emerging as the standard data format for the whole data-related industry because it's recognized across applications and across platforms. Keeping pace with the current advance in technology, Crystal Reports allows you to report off existing XML data and to export your work to XML format.

Access XML data at runtime

Use the Report Designer Component to access XML data streams at run time and pass that information to objects in report applications.

Note: For information on using XML, refer to the new "XML" chapter of the *Crystal Reports User's Guide*.

Export report data straight to XML

Export your report data to XML format quickly at runtime. This allows you to send the data to other eCommerce applications used to read and manipulate information.

New properties have been added to the ExportOptions object to support these new features. These properties are:

- XMLAllowMultiplePages
- XMLFileName

Note: For more information see "ExportOptions Object Properties" in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Improved interactivity with parameters

Using parameters not only enables user-driven reporting, but also increases report performance, especially over the Web. Create multi-purpose reports to ensure that everyone receives the right information quickly.

All of Crystal Reports' web-enabled report viewers have been improved to allow report designers and end users to utilize parameters to their full potential over the Web. Whether users prefer the Java, ActiveX, or DHTML viewers, they can interactively regulate the information they see by accepting default report parameters, picking from listed options, specifying multiple values, or entering ranges.

Customizable DHTML Report Viewer

Customize the Crystal Reports Dynamic HTML viewer to integrate seamlessly with the design of an existing eCommerce web site or corporate intranet. The DHTML viewer exposes the operations of its toolbar (via JavaScript), thereby giving report designers the ability to customize both its workings and its look-and-feel.

The enhanced DHTML viewer is so flexible that users can view reports, drill down on charts, link to on-demand subreports, or quickly export valuable data—all without even seeing that their experience is powered by Crystal Reports.

Display web reports independent of server location

Use Relative URLs to share your reports and their associated report objects on any server, in any directory. Each report remains independent of its actual location, because Relative URLs eliminate dependence upon any particular server or virtual directory.

Extend reporting abilities with additional applications

With Crystal Enterprise Standard, use the ePortfolio, other samples, or any client application designed by your Administrator (using the SDK) to access shared reports.

Additional Web reporting features

HTML and DHTML export

The performance of the HTML and DHTML export has been greatly improved.

Re-distribute the ASP Web Report Server

The enhanced ASP Web Report Server (rptserver.asp) is no longer dependent upon the Web Component Server, so developers can easily distribute their ASP web applications. (Please see "License.hlp" for redistribution rights.)

Facilitate administration with the license manager

Keep track of your Crystal Reports licenses and key codes with the License Manager (available only with Crystal Reports Developer Edition). This new administrative tool makes it easy for you to ensure that your product installation always meets your reporting needs. It also allows you to check the number of concurrent user licenses being used in the ASP environment. Access the License Manager from the Crystal Reports Tools program group on the Start menu.

Please note that the Report Designer Component will work only within the concurrent license limit when used as a reporting server.

Note: For more information see "Licence Manager" in the *License help* (*License.hlp*).

Report Designer Component 8.5 (RDC)

Designed for Microsoft Visual Studio and other COM-based development environments, the feature-rich Report Designer Component 8.5 (RDC) gives developers unprecedented control over report layout and formatting for web and Windows application reporting.

Reflecting the enhanced features of the Crystal Report Engine, the RDC now supports most of the new Crystal Reports features, including PDF export, XML export and data access, alerting, summaries for hierarchical groups, and many more.

Embed a Crystal Reports Designer in your application

Your users can now design or edit reports at runtime with the Embeddable Designer. This easy-to-use ActiveX control provides the full power of Crystal

Reports to your application. The control is placed on a form, or in a container within the form, and displays a new or existing report in design mode. This presents the user with an interactive environment in which to design or edit the report. The Embeddable Designer allows users to:

- add data sources
- add database fields
- create and add formulas
- create and add parameters
- group data
- summarize data
- add charts
- add subreports
- add special fields
- format fields and sections.

Note: For more information see "Embeddable Crystal Reports Designer Control Object Model", and "Programming the Embeddable Designer Control" in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Create and modify Report Alerts

Report Alerts are custom messages created in either the Crystal Reports Designer, the Report Designer Component, or the Embeddable Crystal Reports Designer Control. Report Alerts may indicate action to be taken by the user or information about report data. For more information see Report Alerts in the *Crystal Reports User Guide*.

Two new collections and two new objects have been added to the RDC object model to allow you to create, modify, and monitor Report Alerts at runtime. These collections and objects are:

- ReportAlerts Collection
- ReportAlert Object
- ReportAlertInstances Collection
- ReportAlertInstance Object

Note: More information on the Report Alert collections and objects can be found in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Convert database drivers

The data source your report is pointing to can now be changed at runtime by converting the database driver. You can have this change reflected immedialtely in your application, or save the report, and have the change take effect the next time it is refreshed.

For more information see the ConvertDatabaseDriver method in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Re-import subreports

Subreports can be newly created in the main report or imported from an existing report file. If the subreport is imported from an existing report file, it can be re-imported at runtime using the ReImportSubreport method. When previewed, printed, or exported, a re-imported subreport will reflect all changes made to the formatting, grouping, and structure of the existing report file.

For more information see the ReImportSubreport method in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Summarize on hierarchical data

Make large, complex data sets easier to understand than ever by taking advantage of the new ability to create Subtotals, Grand Totals, and Summary fields for hierarchical groups: to do so, select the "Sum across hierarchy" option, which has been added to the Insert Subtotal, Insert Summary, Insert Grand Total, and Change Summary dialog boxes.

Hieracrchical summaries can be enabled at runtime through the HierarchicalSummaryType property. This property has been added to the following two objects:

- RunningTotalFieldDefinition Object
- SummaryFieldDefinition Object

Note: Morett information on the RunningTotalFieldDefinition object and the SummaryFieldDefinition object can be found in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Export to paginated PDF, and paginated RTF

Export your reports—in whole or in part—directly to the popular Portable Document Format (PDF), and then distribute these reports by email, over the Web, or in print. Alternatively, take advantage of the new Rich Text Format (RTF) exporting feature, which is based on the Crystal Reports Encapsulated Page Format (EPF).

In addition, Crystal Reports now supports page ranged export to PDF and RTF formats.

New properties have been added to the ExportOptions object to support these new features. These properties are:

- PDFExportAllPages
- PDFFirstPageNumber
- PDFLastPageNumber
- RTFExportAllPages
- RTFFirstPageNumber
- RTFLastPageNumber.

Note: For more information see "ExportOptions Object Properties" in the *Crystal Reports Developers Help* (*CrystalDevHelp.chm*).

Crystal Report Viewer

An integral part of the Report Designer Component, the Crystal Report Viewer offers full control over the previewing of reports in both client and Web environments. A new freeze pane feature and improved distribution further enhance the viewer's functionality.

Freeze panes in the Crystal Report Viewer

The ActiveX viewer now has a feature similar to the Freeze Pane feature found in Microsoft Excel. In the ActiveX Viewer, right-click and select Freeze Pane from the shortcut menu to freeze the report from the bottom-right corner of the selected object. Unfreeze using the same menu. Scroll horizontally or vertically, and the report data scrolls as it would in Excel: vertical scrolling scrolls the portion of the report to the right of the selected object (above and below); horizontal scrolling scrolls the portion of the report below the selected object (left and right of the object).

Improve runtime distribution

Distribution of the Crystal Report Viewer (Crviewer.dll) has been simplified.Crviewer.dll is no longer dependent on the Urlmon.dll making for an easier installation of the RDC with your application.

Additional information

Crystal Reports product news

`http://www.seagatesoftware.com/products/crystalreports/`

Crystal Reports demos

`http://www.seagatesoftware.com/products/crystalreports/showme/`

Crystal Enterprise product news

`http://www.seagatesoftware.com/products/crystalenterprise/`

Product information

`http://www.seagatesoftware.com/products/`

Developer Zone

`http://www.seagatesoftware.com/products/dev_zone/`

Online support, samples and tech briefs

`http://support.seagatesoftware.com/homepage/`

Training and consulting

`http://www.seagatesoftware.com/services/`

Seagate Software homepage

`http://www.seagatesoftware.com/homepage/`

Integration Methods 2

This chapter is to gives you an understanding of how integrating Crystal Reports into your applications has evolved. You will also learn the benefits of using the latest technology—represented in the Report Design Component (RDC). This technology will enable you to take full advantage of the powerful Crystal Report Print Engine and create the best applications available today.

Overview

Crystal Reports is the world standard for desktop and web reporting. The Developer Edition includes many innovative features for developers who need to integrate sophisticated reporting into their Visual Basic®, Visual C++®, Visual J++®, Office, Java, Lotus Notes, and Delphi applications.

The methods developers use to integrate reporting into their applications have evolved over time. Industry leaders such as Microsoft have focused their efforts on integrating new technologies into their developer software to make it easier to create powerful applications. As companies such as Microsoft have introduced these advancements, Crystal Reports has kept pace to ensure developers can take advantage of these advancements when using our software.

Note: Visit the Seagate Software Developer Zone web site at `http://www.seagatesoftware.com/products/dev_zone.`
Click Support to get links for finding documentation and knowledge base articles about integrating reporting in your applications.

Integration methods

You can choose to access the Crystal Report Print Engine and integrate Crystal Reports' functionality into your database applications in a variety of ways:

- Report Designer Component Runtime (RDC)
- Crystal Report Automation Server
- Crystal ActiveX® Control (OCX)
- Crystal Report Print Engine API
- Crystal Visual Component Library (VCL).

When they were first introduced, each of these methods represented the latest technology available at that time. Today, you can take advantage of the full power and functionality of Crystal Reports by using the Report Designer Component, our most recent technology.

Method	**Introduced in version**	**Project Reference Name**	**Description**
Report Designer Component	Crystal Reports 7	Crystal Report 8 ActiveX Designer Runtime library (craxdrt.dll)	32-bit only COM object model, dual interface, apartment model
Crystal Reports Automation Server	Crystal Reports 6	Crystal Report Print Engine 8 Object Library (cpeautl6.dll/cpeaut32.dll)	32- and 16-bit COM object model, dispatch only Interface for C developers
Crystal ActiveX Control (OCX)	Crystal Reports 4.5	ActiveX Control (crystl16.ocx/ crystl32.ocx)	32- and 16-bit OCX
Crystal Report Print Engine API	Crystal Reports 3	Crystal Report API Interface, declares encapsulated in GLOBAL32.bas	

The Report Designer Component

The Report Designer Component (RDC) offers a dual interface COM component which is the most advanced technology available for developers. It is an ActiveX Designer that utilizes the reporting power of Crystal Reports. If you're creating applications in popular environments such as Visual Basic, Visual C++, Visual InterDev® and Office 2000 (VBA), you can use the RDC to efficiently create the most powerful reporting functionality in your applications

Although the RDC is our latest integration technology, Crystal Reports 8 supports the older integration methods (32-bit only). We recommend, however, that you begin using the newest and best reporting technology available—the RDC—to create powerful reports that support the entire range of functionality available within Crystal Reports.

To better understand why the RDC offers the best solutions, it's important to understand its architecture and how it relates to different types of developers.

The RDC consists of three main components:

- **Report Designer:** for Visual Basic and Office 2000 (developer edition with VBA 6.0) developers, this component lets you create, view and modify reports directly in the Visual Basic or VBA Integrated Development Environment (IDE).
- **Automation Server:** an extensive object model with hundreds of properties and methods you can use to program a report.
- **Report Viewer:** used to preview reports on screen, the Report Viewer gives you great control and flexibility over the viewed report. It's a lightweight ActiveX control that contains an extensive object model. The viewer is customizable and allows you to add a custom toolbar if needed.

Visual Basic and Office 2000 (VBA) Developers

Because the report designer is an ActiveX designer, it is easy for Visual Basic and VBA developers to use:

- The report designer is closely integrated with Visual Basic 5.0 and 6.0 and VBA 6.0, and it allows you to open, design, and customize reports directly from the Visual Basic or VBA IDE.
- Report Experts guide you in creating the report you need, allowing you to quickly and easily design reports.
- You can use familiar Visual Basic code to create even more advanced functionality.
- The RDC is an easy way to give your applications powerful reporting capabilities.

Visual InterDev Developers

Many Visual InterDev developers are experienced with using Visual Basic, and are therefore familiar with integrating reports into database applications. Developers can use the same RDC automation server and Report Viewer in Visual InterDev as they use in other COM-based environments such as Visual Basic and Visual C++ to access the features of Crystal Reports. This provides one object model that offers the same control over the report engine in web applications as well as Windows applications. Developers can save time since they can build a report once and move it to either the desktop or the Web. Report Integration controls then give Visual InterDev developers an easy way to integrate reports into ASP pages.

Visual C++ Developers

If you use a development environment such as Visual C++ that supports COM, you can:

- create reports in Crystal Reports
- use the RDC's automation server and viewer to manipulate and preview those reports at runtime.

You will save time writing code, and can use the RDC to integrate reporting in the most logical and efficient way.

Evolution of Integration Capabilities of Crystal Reports

The integration technology available to developers has progressed in parallel with advancements in the developer market. Crystal Reports has quickly adopted the leading concepts and architecture implemented for Visual Basic, Visual C++, and other development environments. The following is an overview of how the application integration methods offered in each version of Crystal Reports have evolved over time.

Crystal Report Print Engine API

Since Version 1, Crystal Reports has exposed the Report Engine functionality via a C API that you could use to make direct Report Engine calls.

Using these direct Report Engine calls you can access more functionality than offered with the OCX. However, because it was designed for C/C++, it wasn't easy to use the API inside Visual Basic and some of the API wasn't available because of Visual Basic limitations in creating C style structures.

As a Visual Basic developer, you needed to add more code—and in many cases complex code with multiple parameters—into an application to execute a report. This was especially time consuming if you were unfamiliar with API calls.

If you have used the Report Engine API, the RDC is appealing because it offers additional functionality, and is a much easier way to experience the power of the Crystal Report Print Engine.

The Crystal Report Engine API is only available in standalone versions of Crystal Reports (i.e. version 8). No new features will be added in future releases of Crystal Reports.

The Crystal ActiveX Control (OCX)

The Crystal ActiveX Control (OCX) is one of the oldest development technologies available to developers in Crystal Reports. When OCX was first released in Crystal Reports 4.5, it was based on the latest ActiveX technology from Microsoft, thus offering state-of-the-art reporting for Visual Basic developers. Today, however, its capabilities pale in comparison to the power and flexibility available in the RDC.

- The OCX control lacks programming depth and complete Report Engine functionality.
- While the Crystal Report Engine provides features that are conducive to a hierarchical object model—where each report is broken down into sections with objects—this type of hierarchy isn't represented in an OCX control. At the time of its original development, the OCX revolutionized report integration because it provided developers with a graphical interface to integrate existing reports. But all of the Report Engine features can't be represented in the flat model allowed by the OCX.
- In addition, the OCX is a wrapper around the Report Engine, which means it is less efficient and requires the distribution of more runtime components than using the RDC.

Today, the OCX supports only the smallest subsets of Report Engine functionality—the most common properties and methods. Although it's supported in newer versions, no new features have been added since Crystal Reports 6.

If you're a Visual Basic developer who has used the OCX, you will find the RDC easier to use inside the familiar Visual Basic IDE and by taking advantage of Report Creation experts.

Crystal Report Automation Server

Based on the Crystal Reports 6 Object Model, the Crystal Report Automation Server uses a hierarchical object model, unlike the "flat" model of the OCX.

- The Automation Server provided the first object-oriented interface to the Report Engine.
- It was also the first object model in Crystal Reports that allowed Visual Basic developers to access all the Report Engine features, something C/C++ developers had enjoyed since the beginning.

It was built around the Report Engine DLL. It behaved as a translation layer, taking Visual Basic code and converting it into Report Engine calls. Because of its inability to directly access the Report Engine, the Crystal Report Automation Server wasn't as efficient as it could be.

We've ensured backward compatibility in Crystal Reports 8 to allow you to leverage applications created in the previous version. However, the Crystal Report Automation Server no longer exposes all of the events and properties of the Report Engine in Crystal Reports 8. Because the Report Designer Component is COM-based, and an evolution of the Crystal Reports Automation Server, you will find it easy to migrate from the Crystal Reports Automation Server to the RDC.

Report Designer Component (RDC)

The RDC—introduced in June 1998 as a component of Crystal Reports—represents a major reengineering of the Crystal Report Print Engine. Unlike any of its predecessors, the RDC is not a wrapper; it exposes all Report Engine objects directly without any translation. The RDC is based on the same object model as the Crystal Report Automation Server. Because it is not a wrapper, the RDC is a much more efficient COM object that supports features such as dual interface, providing a more efficient way of making calls to the Report Engine.

The primary advantage of using the RDC over other developer tools within Crystal Reports is its code writing and formatting capabilities in popular development environments such as Visual Basic. The RDC provides events that enable you to manipulate the report at runtime. Within these event handlers, you can access text or field objects to modify the output of a report based on user input. Setting text in text objects or field objects, or dynamically changing pictures in picture objects, are added features unique to this interface.

The Best Tool for Your Needs

We recommend that you use the RDC to take advantage of the best functionality and features available. While we support applications created using other tools like the OCX and the Crystal Report Print Engine APIs, the RDC offers more power and is easier to use.

New features of the Crystal Report Print Engine are only available through the RDC. If you'd like to integrate them into your applications, you must change the code. Links to resources (technical briefs and tutorials) to help you migrate to the RDC are located on the Seagate Software Developer Zone web site.

Visit the Seagate Software Developers Zone at
`http://www.seagatesoftware.com/products/dev_zone`
to find information on the resource that best suits your needs. Chapter 9, "Migrating to the RDC from the OCX" on page 119 of this guide shows you how to move quickly to the RDC from the OCX.

The RDC is the premium development method, and it will continue to be enhanced for developers. The latest version of the RDC is included in Crystal Reports 8.5.

Introducing the Report Designer Component 3

This chapter introduces the Report Designer Component (RDC) and explains it's architecture along with some possible development scenarios. It also provides additional sources for help and information on the RDC, including sample applications that present the RDC in real-world scenarios.

The Report Designer Component

The Report Designer Component (RDC) is a powerful solution for Visual Basic developers who quickly and easily integrate reporting into their database applications. It's an ActiveX designer object that packs the reporting power of Crystal Reports into a lightweight add-in for Visual Basic 5.0 or 6.0. You can open, design and customize reports within the Visual Basic IDE. In addition:

- Intuitive Report Experts make it flexible and efficient to connect to data and integrate powerful reports into applications.
- With hundreds of report properties, methods and events, you have complete control over your report designs, using familiar Visual Basic code.
- Report distribution is simplified through a small component count and free runtime.
- Reports can be packaged within the applications' executable or stored outside the application in the traditional (.rpt) format.

Note: Get the latest news about the Report Designer Component from the Seagate Software Developer Zone web site at
`http://www.seagatesoftware.com/products/dev_zone`

Design time

At design time, the Report Designer Component provides a user interface that closely integrates with the Visual Basic IDE. Through the user interface, you design and manipulate reports and report data. This interface includes events that can be directly programmed from within Visual Basic.

The Report Designer Component uses the Active Data Driver for connecting to ISAM, ODBC, and SQL databases through Data Access Objects (DAO), Remote Data Objects (RDO), ActiveX Data Objects (ADO), and Data Environments (Visual Basic 6.0 only). You can design the data set from within Visual Basic, then apply it to the report contained by the Report Designer Component.

When working in Visual Basic, you will often need to use the Report Viewer for ActiveX as a user interface to display reports. The Report Viewer is an ActiveX control that you can drop to a standard Visual Basic Form. The Report Viewer is where your report is displayed at runtime.

Runtime

The user interface provided by the Report Designer Component at design time does not appear in your application at runtime, or when it is compiled into an executable file. Instead, the Report Designer Component is accessed directly by your Visual Basic code. New for version 8.5 is the Embeddable Crystal Reports Designer Control. This allows you to provide your users with the ability to design or edit Crystal Reports within your application using the same interface as the design time control. However reports can be opened and saved only in the Crystal Report format (.rpt) and not in the ActiveX Designer format (.dsr) of the design time control.

The Report Designer object model provides a complete object hierarchy for direct manipulation in Visual Basic. The object model can be accessed through both the RDC runtime engine and the RDC design and runtime engine. See CR Automation Server for more information.

The Active Data Driver is also available at runtime, through the Report object of the Report Designer Component object model, and can be assigned a new set of data based on user interaction with your application. You design a Recordset or Resultset object in Visual Basic using the DAO, RDO, or ADO object model, and pass it to the report.

Finally, the Report Viewer takes center stage at runtime, connecting to the Report Designer Component and displaying the embedded report. With careful design and placement on the Form, the Report Viewer appears as a window inside your application.

RDC Architecture

If you aren't familiar with the RDC, you should note that it consists of five components. Together, these components enable you to create, design, edit, program, and preview, print, or export your reports:

Report Designer

The Report Designer is integrated tightly within the Visual Basic 5.0 and 6.0 IDE, enabling you to create, view, and modify reports without leaving Visual Basic. This component was specifically created for Visual Basic developers.

Automation Server

There are two automation servers provided with the RDC. The Crystal Reports ActiveX Designer Runtime Library (craxdrt.dll)—known as the RDC runtime engine, and the Crystal Reports ActiveX Designer Design and Runtime Library (craxddrt.dll)—known as the RDC design and runtime engine. Both offer the same extensive object model with hundreds of properties and methods for you to use to manipulate the report at runtime. See "Unification of the RDC object Model" in the *Crystal Reports Developers Help (CrystalDevHelp.chm)* for more information on when to use each automation server.

Crystal Report Viewer

The Crystal Report Viewer is an ActiveX control that you can use to preview reports on screen. The viewer is loaded with customization options to give you great control over your interface and over the viewed report itself.

Embeddable Crystal Reports Designer Control

The Embeddable Crystal Reports Designer Control (Embeddable Designer) is an ActiveX control that you can use to allow your users to design and edit Crystal Reports in your application at runtime.

Component	**Description**
Crystal Report Designer UI Component (craxdui.dll)	A COM (Component Object Model) component that provides the user interface at design time for the user to interact with and create or modify the report.
Crystal Report Designer Design and Runtime Time Component (craxddrt.dll)	An underlying COM component that provides services for the user interface component. The component also encapsulates all of the report objects and is responsible for all of the data processing and report layout. Use this component with client based applications that incorporate the Embeddable Designer.
Crystal Report Designer Run Time Component (craxdrt.dll)	The component that encapsulates all of the report objects and is responsible for all of the data processing and report layout. Use this component with any Client based or server side application that does not incorporate the Embeddable Designer.
Active Data Driver (p2smon.dll)	A data access driver that provides access to various types of object data sources including DAO, RDO, and ADO.
Crystal Report Viewer for ActiveX (crviewer.dll)	An Active X control which can be drawn on a form and manipulated at design time. It provides a rich object model used to modify user interaction with the report at runtime. This component is required only if a developer wants to provide on-screen display of reports at runtime.
VB Form	The Crystal Reports Report Viewer Control must be embedded on a Visual Basic Form in order to display the report on screen. The Create Report Expert can automatically add a Form with the Report Viewer embedded to the project when you finish designing a report with the Expert.
Data Set	One of the following: ■ Data Access Object (DAO) Recordset ■ Remote Data Object (RDO) Resultset ■ Active Data Object (ADO) Recordset ■ VB Data Environment ■ Crystal Data Object (CDO) ■ Crystal Data Source Type Library object ■ ODBC Direct These objects do not need to be valid at design time. For example, you could construct a report template at design time without the data being available. This is handled through "Data Definition Files" on page 45. However, the data set objects must be present and valid at runtime to generate a report.

Combinations of RDC Components

The three RDC components can be used together or individually, in any combination. The following table identifies which of the components you need to use in your project:

Using the Report Designer with the Automation Server	■ VB and VBA only ■ for designing reports within IDE and manipulating the reports with code ■ not necessary to preview the reports ■ reports can be printed or exported ■ use the Crystal Formula Language, VB or VBA code for expressions in the report.
Using all three RDC components	■ VB and VBA only ■ want to design reports within IDE and manipulate the reports with code ■ reports can be previewed, printed or exported ■ use the Crystal Formula Language, VB or VBA code for expression in the report.
Using the Automation Server and the Crystal Report Viewer	■ VB and VBA, C++, Visual InterDev, Visual J++ ■ want to design reports using Crystal Reports, preview from application, and manipulate the reports with code ■ reports can be printed or exported ■ use the Crystal Formula Language, VB or VBA code for expressions in the report.
Using the Automation Server, Embeddable Designer, and the Crystal Report Viewer	■ VB and VC++ ■ want to allow your users to design or edit their own Crystal Reports ■ reports can be saved, printed, exported and previewed. ■ use the Crystal Formula Language, VB or VBA code for expressions in the report.
Using the Automation Server by itself.	■ VB and VBA, C++, Visual InterDev, Visual J++ ■ want to design reports using Crystal Reports and manipulate the reports with code ■ reports can be printed or exported ■ use the Crystal Formula Language, VB or VBA code for expressions in the report.

Adding the RDC to your project

The installation program will attempt to add the menu item **Add Crystal Reports 8.5** to the Project menu of the VB IDE when you install the RDC. If that menu item appears, you simply select it to add the RDC to your project.

If the menu item does not appear on the Project menu, you will first need to add it manually, and then you can add the RDC to your project.

To add the RDC to your project

1. Choose Components from the Project menu.
2. Click the Designers tab in the Components dialog box when it appears.
3. Select Crystal Reports 8.5 on the Designers tab. This will add the menu item to the Project menu.
4. Finally, select Add Crystal Reports 8.5 from the Project menu.

The RDC is now ready to use within the VB IDE.

How to Get Additional Information

Seagate Software supplies an extensive set of tools to help you master the RDC quickly and easily. These include:

"Crystal Reports User's Guide" on page 20

"The Seagate Software Web Site" on page 21

"Object Browser" on page 21

"Properties Window" on page 21

"Sample Reports" on page 22

"Sample Applications" on page 22

Crystal Reports User's Guide

Since the capabilities of the RDC mirror the capabilities of Crystal Reports, you can use the *Crystal Reports User's Guide* to learn about the RDC's powerful reporting capabilities. For example, if you want to learn the details of the Select expert, how to link a subreport to its parent report, or just more general information on report creation, you should refer to the User's Guide. This documentation is available in printed form as well as a searchable electronic (PDF) form on the CD. You can find the User's Guide (Usergde.pdf) in the Docsfolder on the CD.

Help System

Crystal Reports comes with an extensive help system. The help files contain all of the information from the printed documentation and considerable additional information as well. You can find reference information about all of the properties, methods and events in the RDC, all the calls and structures in the API, and much more in the Help system (CRRDC.hlp and CrystalDevHelp.chm).

The Seagate Software Web Site

Seagate Software provides a wide range of developer support information and tools on its web site at
`http://www.seagatesoftware.com/products/dev_zone`

The web site offers these support tools and more:

- downloadable files and product updates
- developer samples
- a comprehensive knowledge base showing solutions and workarounds to a wide range of technical problems
- technical briefs, detailed documents that discuss complex issues and/or explain how to use various sophisticated product features
- FAQs: quick answers to your most common questions
- Release Notes: details about known issues, product features, enhancements, and fixes.

The web site is updated continually so you will always be able to find current information to help you with your development challenges.

Object Browser

A source of support for many VB programmers is the VB Object Browser. Use the Object Browser to navigate the RDC automation server object model (CRAXDRT), the Report Viewer object model (CRVIEWERLibCtl), the various objects used in your project (Project n), and the VB and data object models. The Object Browser introduces you to the hierarchy of the object models, and provides you with useful information about the various classes and members in the object models.

Properties Window

The RDC exposes each object in an RDC report. This means you can view all of the design time properties and options in the Properties Window of the VB IDE. Simply select an object in the report and you can see and set its various properties visually.

Sample Reports

The program includes many sample reports you can use to learn sophisticated reporting techniques. With a full installation, you can find these samples in the \Crystal Reports\Samples\Reports folder.

Sample Applications

Crystal Reports contains over 18 Visual Basic sample applications. These applications show you how to use the RDC in real-world situations.

With a complete installation, you will find the sample applications in the \\Seagate Software\Crystal Reports\Samples\En\Code\Visual Basic folder. There are samples for C++ and the Web in the \Code folder. The sample applications provide you with one of the most direct and tested methods for learning the RDC.

The applications contain usable code that can be modified and plugged into your own programs. They also present great ideas for using the RDC. By studying these ideas, you may be able to generate new business by offering clients sophisticated capabilities.

Visual Basic Samples

The following sample applications have been created for Visual Basic developers:

Simple Application

Most of the sample applications focus on solving a single problem or a small set of related problems. If you're a first time user of the RDC, you'll find the application called Simple Demo a good place to start.

ADO Connection Methods

This application demonstrates the two new methods to add ADO data sources to a report:

- AddADOCommand takes an active ADO connection and ADO command object as parameters. In this example, a new connection is created to a database, and the resulting recordset is assigned at runtime to the report.
- AddOLEDBSource takes as parameters the name of an active ADO connection and the name of a table that can be accessed through that connection. In this example, the connection setup through the VB Data Environment is used as a data source.

Both methods can be used with either the VB Data Environment or another data source created at runtime.

Note: Notice in the Designer that there is no data source attached to the report at design time.

Change Runtime Location of OLE Object

This application demonstrates how you can set the location of an OLE object at Runtime using the SetOleLocation command.

This application takes three different OLE objects:

- a bitmap
- part of an Excel spreadsheet
- part of a Word document.

This application cycles through them so a different object gets printed each time the Details section gets formatted.

Embeddable Designer Control

This application demonstrates how to include the Embeddable Crystal Reports Designer Control in an applicaion. It shows how to add a new or existing report to the Embeddable Designer, how to display the report in the Crystal Report Viewer, and how to refresh the report in the Crystal Report Viewer so it displays any changes made in the Embeddable Designer.

Employee Profiles

This application demonstrates how you can set and format the text in report text objects at runtime.

Inventory Demo

The Inventory Demo is a sample application that displays a report. It shows you how to use the Report Initialize section for report level manipulation. It also includes how to change the BackColor of the report Details section conditionally for those items that need to be ordered.

Load Picture Demo

The Load Picture application shows you how to load a picture into the report at runtime. It also shows you how to change some of the values and properties of some of the text objects in the Details section of the report at runtime.

The program also shows you ways to implement calculations and calculated fields in the RDC. The two methods are:

- using the Crystal Formula Language
- using VB code in the Section_Format event procedure.

This example describes how to perform calculations in the Section_Format event using VB.

Microsoft Office Samples

Visit the Seagate Software Developer Zone web site at
`http://www.seagatesoftware.com/products/dev_zone`

To search for Microsoft Office sample applications click Support, and then click the Download Files and Updates link.

No Data in Report Event

This application demonstrates the new NoDataInReport event.

The "No Data" event is triggered when the data set in the report is empty. In this sample, the code applies a filter to the data that causes the report to have no data. It then traps the NoDataInReport event and displays a message box that allows the user the option to view the report without any data or just to view a blank page.

Printer Settings

This application demonstrates how to change the report printer settings at runtime using code.

- There are four new report properties that allow you to easily retrieve and set PaperSize, PaperOrientation, PaperSource, and PrinterDuplex for the report printer options. All of these are demonstrated.
- There is also a new method called PrinterSetup that provides a Windows standard printer setup window to allow the user to change the printer properties directly at runtime. This is demonstrated as well.

The two methods are independent of each other. For example, changing the Windows standard printer setup will not alter the report printer settings and vice-versa. These new properties and the new method give you much more control over how the report is printed.

Report Object Creation API

This application demonstrates the use of the new Report Object Creation API.

The Creation API functions allow runtime creation of report objects, including text, database field, unbound field, chart, special, box, cross-tab, BLOB field, line, picture, summary, and subreport objects.

- You can add these report objects at runtime to an existing report that you created in the RDC.
- You can also create a blank report at runtime and add these objects to that blank report. This sample demonstrates the second method.

All properties that are normally available for each object at runtime are also available for the objects you create.

This sample shows you how to create text, database field, unbound field, summary, picture box and line objects. For information about more advanced uses of the Creation API, please see the Pro Athlete Salaries sample application.

Report Variables

This application demonstrates the use of the new Report Variable capability.

The RDC now includes Report Variables. These are “smart” variables that can be used to maintain state when you are coding items such as running totals based on a Section_Format event. These Crystal variables can be used in your VB code. They understand what’s going on inside the report formatting process at any given time, and they enable you to do more of your report-oriented coding exclusively in VB.

This sample describes how to use Report Variables to calculate totals based on only part of the data. Since Report Variables accept any valid value from the VB environment, you can use values that only exist at Runtime to change the formatting of your report.

In this sample, there are three columns based on a database: one column of currency values, one column of integer values, and one column of non-integer values.

- The first column changes to a blue color when the sum of the values in the column is greater than a random value. A total is displayed in the lower part of frmMain showing the total of the values colored blue.
- Records in the second and third columns change color when the sum is greater than a different random value. The sum is then reset. This can be useful if you wish to start and restart a totalling operation in the middle of a report.
- A String Report Variable is used to build a string consisting of the first digit of each green highlighted value. You could use this method to easily extract individual string values from a report field.

Note: Report Variables can only be set and retrieved while the report is formatting.

Report Wizard

You can use the Report Wizard with the Report Creation API to enable users to create entirely new reports at runtime and then view or save the report. When you implement the Wizard, you save yourself the necessity of writing “create” code for each of the objects your users put into their new report. This application shows you how easily you can implement the Wizard in your code.

You can also use the Report Wizard as a starting point for creating a custom report wizard that allows you to set your own defaults for report creation.

Search and Select Experts

This application shows you how to control the Search and Select Wizards in the Report Viewer using code.

The Search and Select Wizards allow advanced filtering and highlighting of data displayed in the ActiveX viewer. In this sample, two different methods of using the Search and Select Wizards are demonstrated:

- Two command buttons on the main form display a search or select wizard with predefined fields and ranges already set for the user. The only displayed fields and ranges are those that have been added at runtime.

- The Search and Select Expert buttons on the top right part of the viewer are over-ridden, allowing you to specify default criteria to search for. Alternatively, the default search and select experts can also be displayed.

Simple Demo

This simple demonstration presents an introduction to the Report Designer. Many of the basic methods used in manipulating and creating reports are discussed in the code comments. This is a good starting point if you are unfamiliar with the RDC.

Unbound fields

This application shows you how to create a report template using the new Unbound Fields capability. It also shows you how to bind the fields at runtime to a data source that you also create at runtime.

In this sample, there are five unbound fields created in the report. Notice that there is no data source added to the report at design time at all—the data source is created and added to the report at runtime.

You can also use Unbound fields in formulas, although this sample does not demonstrate this use.

Unbound fields give you great flexibility in your programming. They even allow you to create a single report template that can be bound to a variety of data sources at runtime in response to user input. This enables you to create a wide range of custom reports all from a single template. See the Pro Athlete Salaries application for a demonstration of this capability.

Viewer

This simple application shows you how you can create your own Report Viewer toolbar and manipulate the ActiveX viewer using your own custom buttons.

Viewer Runtime options

This application demonstrates the many ways you can manipulate the Report Viewer at runtime. It also shows you how to implement the use of the Select Expert and the Search Expert in the Viewer.

All of the check boxes on the main form correspond to viewer properties that you can change at runtime. In this sample, there are more than 21 viewer properties that are manipulated. Note that to enable the Select Expert and Search Expert, you must reference the Crystal Select Expert OLE control module and you must place the Select Expert Control on the form.

The Select/Search Expert is essentially invisible on the form and is located immediately to the left of the About Sample Command Button in this sample.

Complex Applications

The following three applications are large and complex. They each demonstrate many of the RDC capabilities.

Xtreme Mountain Bikes

This application was included with Version 7. It has been updated in this version to showcase many of the new RDC capabilities.

Pro Athlete Salaries

This application shows a very sophisticated use of the Create Report API, Unbound Fields, and a previously-undocumented method for using the Tables.Add method. This report is worth studying because it demonstrates the powerful things you can do with the RDC through code alone.

Note: The Designer does not appear in the Project Explorer. This is because the report is actually a template report created at runtime using Unbound Fields and the Create API, and the fields are bound at runtime to a custom recordset generated at runtime based on user input.

First Class Hotels

The sample demonstrates a number of Crystal Reports features that are new to version 8, as well as possible advanced uses for some existing features.

The main form in the sample, frmMain, is used for managing the reservations. Some features include:

- The report displayed in the window is automatically updated whenever a reservation is added, the customer name changed, or the date range changed.
- Double-clicking on a customer name or room number in the report will bring up a form containing information about that customer or room.
- Clicking on one of the two miniature notepad icons in this window will also cause this same information report to be displayed.
- The billing report that can be displayed by double-clicking the report, or clicking the notepad icons, only displays information between the start and end dates shown at the bottom of the reservation form.

The Room form, frmRoom, contains these features:

- Allows adding/removing of hotel room numbers.
- Clicking on the miniature notepad icon in this window will cause the information report to be displayed. The information will be displayed for all dates that the room has been used and is not restricted by a start or end date.

The Price form, frmPrice, contains these features:

- Allows adding/removing of hotel room prices.

- Customer: Allows adding/editing/removing of customers.
- Clicking on the miniature notepad icon in this window will cause the information report to be displayed. The information will be displayed for all dates that the customer has stayed at the hotel and is not restricted by a start or end date.

This application uses many of the methods from earlier releases as well as this release. Here is a list to help you determine if this application has the answers you're looking for (* denotes new in version 8):

Crystal Reports Viewer Object

Methods:	**Properties:**	**Events:**
PrintOut	EventInfo.Text	DblClicked
Refresh	EventInfo.Type	
ReportSource		
ViewReport		

Crystal Reports Report Object

Methods:	**Properties:**	**Events:**
*AddReportVariable	Get and Set PaperSize	*BeforeFormatPage
*GetReportVariable	Get and Set PaperOrientation	*EndFormatPage
*SetReportVariable	*Get and Set PrinterDuplex	Initialize
RecordSelectionFormula	*Get and Set Papersource	*NoData
*SetUnboundFieldSource	Database.Tables().Location	Section_Format
	LeftMargin	
	RightMargin	

Quick Start for using the RDC 4

This chapter helps you get started using the Report Designer Component (RDC) in Microsoft Visual Basic. Three examples provide you with the basic skills needed to use the RDC. You will find a high-level explanation of the steps required to add the RDC to a project, a tutorial on adding an existing report to the RDC, and a tutorial on creating and modifying a new report.

Overview

Using the RDC will add powerful reporting capabilities to your application, and you can do it all from the Visual Basic IDE.

Note: For information on using the RDC with other development environments, See Chapter 10, "Working with Visual C++ and Visual InterDev" on page 127.

The following three sections were designed to get you started with the RDC:

"Bullet Point Quick Start" on page 30.
If you only need minimal instructions for using the RDC, this section is for you. It includes a higher level, bulleted explanation about how to create, view, and manipulate a report through the RDC.

"Open an existing report" on page 31.
If you have created a report in the standalone version of Crystal Reports and you want to open it through the RDC and preview it in the Report Viewer, you will find this section useful. It includes a lower level, step-by-step approach.

"Create and modify a report" on page 33.
If you want to create, preview, and manipulate a report through the RDC, you will find this section extremely useful. This is also a lower-level, step-by-step approach. It mirrors the procedure of the Bullet Point Quick Start but describes each of the bullet points in more detail.

If you have questions that are not answered by these samples, please see the Sample Applications at \Crystal Reports\Samples\Code.

Bullet Point Quick Start

To add reporting capabilities to your application using the RDC:

- Add the RDC to your project
  - With a complete install, the option Add Crystal Reports 8 appears on the Project menu in the VB IDE. Select Add Crystal Reports 8.
  - If the option does not appear, select Components from the Project menu and check the Crystal Reports 8 option on the Designers tab. Then select Add Crystal Reports 8 when it appears on the Project menu.
- Select a data source

  When you add the RDC to your project, the program presents the Crystal Report Gallery dialog box. If you're not familiar with Crystal Reports, choose the Using Report Expert option and choose Project data from the Data tab in the Standard Report Expert when it appears. This enables you to select the newest technology, the Active Data recordsets for building your report. Select one of the ODBC(ADO) data sources from the combo box, add a new data source, or supply a connection string for an ADO OLE DB connection.

- Create a report

  From the Standard Report Expert, you can build your report by adding fields, setting up grouping and totalling, specifying your sort options, adding charts and selection criteria if you wish, and even adding a professionally designed look. You can do that by using the various Expert tabs.
- Add the Report Viewer to your project

  When you select Finish in the Standard Report Expert, the program automatically adds a form to your project and installs the Report Viewer on the form. It also writes the necessary code for displaying your report in the viewer and resizing the viewer whenever the form is resized.
  - If you don't want the Viewer installed automatically, or if you don't want the Viewer form to be your start-up form, you can change those actions in the Default Settings dialog box.
  - You get to the Dialogue by right-clicking in the RDC and selecting Designer and then Default Settings from the shortcut menu when it appears.
- Create a UI for users to modify the report at runtime

Now that you have a report, you may want to manipulate it with code based on user input. If you haven't done so already, here is the point where you could build the UI for the user using Visual Basic.

- Write the code that does the modifications

  Now you can write the code that ties the users UI selections to the report.
  - Each object in the report is exposed in the Properties window of the VB IDE.
  - Double-click any object and you can write code for the object.
  - Double-click on a section of the report and you can write code that affects the section and the objects in the section.
  - By referencing an object name in your UI code or the object index through one of the many Collections, you can manipulate that object directly based on user input.
- Preview, print, or export the report.

  When you have completed the above, you can run the application and preview, print, or export the report using the UI controls you've set up to do it.

Now you have integrated the most powerful reporting tool into your application.

Open an existing report

In this example, you will open an existing report file in Visual Basic. This will demonstrate how to import existing reports (*.RPT files) and enable you to become familiar with the Report Designer Component user interface. This tutorial uses Crystal Reports 8.5 and Microsoft Visual Basic 6.0.

Ensure that you have completed the installation of the Report Designer Component files and make note of the installation directory if you did not accept the default selection.

To open a report directly within Visual Basic 6.0

1 Open Visual Basic and create a new Standard EXE project by selecting Standard EXE from the start up dialog or selecting it from New Project under the File menu.

2 Add the Report Designer Component to Visual Basic if not already added during the installation process.

3 From the Project menu, select Components.

4 Click the Designers tab and check Crystal Reports 8. Click Apply and then click Close. The Report Designer is now available in this project and any projects you create in the future.

5 Now we need to insert the Report Designer into the project form. On the Project menu, point to More ActiveX Designers and then click Add Crystal Reports 8.

6 The Crystal Report Gallery appears displaying the different types of Report Experts that are available. Since you will be opening an existing report file, click From an Existing Report. Click OK.

7 Browse to the report called World Sales Report at \Crystal Reports\Sample\Reports\General Business\. Click Open. Depending on your setup, you may be presented with a dialog that asks you about adding a form at runtime. Click OK.

8 The Report Designer Component is added to your project and the report layout is displayed in the design window. Report files created in any version of Crystal Reports can be imported in this manner. Before you run the report, right-click on Form2, and select View Code. You should see Visual Basic code that looks like this:

```
Dim Report As New CrystalReport1
Private Sub Form_Load()
          Screen.MousePointer = vbHourglass
          CRViewer1.ReportSource = Report
          CRViewer1.ViewReport
          Screen.MousePointer = vbDefault
End Sub
Private Sub Form_Resize()
          CRViewer1.Top = 0
          CRViewer1.Left = 0
          CRViewer1.Height = ScaleHeight
          CRViewer1.Width = ScaleWidth
End Sub
```

This default code, inserted by the Report Designer Component, will point the runtime Crystal Report Viewer at the report to display the results. This makes it easy to flip between the report design window and the finished report. You can add to or modify this code, but for now we'll just view the report.

9 From the Run menu select Start (F5) or click on the Start button on the Visual Basic toolbar. After a few seconds, you will see a form displaying the finished report in the Crystal Report Viewer. You can resize the form by dragging the lower right hand corner of the Viewer. You can save this project if you like, but it will not be required to complete the steps in the next section.

Now that you have opened and viewed a report, feel free to go back and explore some of the right-click and property settings in the Report Designer window. You may also want to browse the CRViewer class of the CRVIEWERLibCtl object to see some of the properties and methods that you can use to customize the appearance of the viewer at runtime.

Create and modify a report

In this example, we'll create and modify a simple report using the Report Designer Component. This will guide you through the basic steps in connecting to a data source and creating a report. The report you will be creating is a Sales by Region report based on a Microsoft Access database. We'll use the Report Experts to help create the report quickly and easily, then we'll modify some of its properties and make it more interactive by adding some event-driven code.

This exercise only touches on a few of the hundreds of properties and methods available but will give you an idea of how simple it is to fully integrate and customize reports in your Visual Basic projects.

To Add the RDC to your project

1 Open Visual Basic and create a new Standard EXE project by selecting Standard EXE from the start up dialog or selecting it from New Project under the File menu.

2 Add the Report Designer Component to Visual Basic if not already added during the installation process.

3 From the Project menu, select Components.

4 Click the Designers tab and check Crystal Reports 8. Click Apply and then click Close. The Report Designer is now available in this project and any projects you create in the future.

5 Now we need to insert the Report Designer into the project form. On the Project menu, point to More ActiveX Designers and then click Add Crystal Reports 8.5. (In some environments, Add Crystal Reports 8.5 will appear as an option itself on the Project menu.)

To select a data source

1 Check Using Report Expert and select Standard. Click OK. The Standard Report Expert appears. This window allows you to select from either Project or Other data sources. Typically, you use a project-based data source. The Other option enables you to use the native data drivers included with Crystal Reports.

2 We'll connect to the ODBC data source using ADO. Click the Project button.

3 The Select Data Source window appears with ODBC selected by default. Select Xtreme Sample Database from the list of data sources. To see which options are available, click Advanced. The Advanced Options dialog appears. Click OK to accept the default of connecting using ADO.

4 Now that you've specified the data source and connection method, you need to specify which tables to use. Click Next in the Select Data Dialog. This brings up the Select Recordset dialog.

5 Select Customer from the object list and click OK to return to the Data Tab. The item *ado* should be displayed in the Tables list box.

To create a report

1 Now that you've selected the database and the table, you need to specify the fields to include in the report. Click Next to move to the Fields tab.

2 Select the Customer Name database field from the Available Fields list, then Click Add to add the field into the Fields to Display box. Do the same for the Last Year's Sales, City and Region fields.

3 Click on the Group tab and select the Region field, then click Add to add the field into the Group By box. Do the same for the City field. This will group your data by Region and, within each region, by City. By default, the Sort Order for the Region and City Fields is in Ascending Order.

4 Click the Total tab. Because Last Year's Sales is the only numeric field in the report, the Report Expert automatically selects it for totaling.

5 Click the Top N tab. For the Region Tab, choose Sort All Groups based on Sum of Last Year's Sales. Do the same for the City tab.

6 Click the Chart tab and click the Pie in the Chart Type list on the Chart | Type tab. Select the pie chart with 3D visual effect.

7 Click the Chart | Data tab. The Report Expert automatically selects to create the chart based on the sum of Last Year's Sales.

8 Click the Chart | Text tab. In the Title box, type "Sales by Region".

9 Finally, to give the report a professional look, go to the Style tab in the main set of Expert tabs and select the Executive, Trailing Break style. Click Finish. The RDC creates your report.

To add the Report Viewer to your project

1. The Report Expert presents you with the option of adding a form with the Crystal Report Viewer control and setting this as the start-up object. Click OK to accept the defaults. The Expert will format the report and display it in the design window.
2. Click the Start button on the Visual Basic toolbar or press F5 to run your project. After a few seconds, you will see a form displaying the finished report in the Crystal Report Viewer.

This exercise has shown the basic steps for creating a new report. Although you don't always have to use the Report Experts, they make connecting to your data source and creating the initial report fast and easy. You can then alter the look and feel of the report using the Report Designer window. Common tasks like field formatting, adding text and modifying field positions can be accomplished by dragging fields or altering their properties. These can be set using Visual Basic code or in the Visual Basic object properties window.

Create a UI for users to modify the report at runtime

When you add report capabilities to your application, you provide the user with a UI for customizing their reports (setting the date range for the report or deciding whether they want to see a detail or summary report for example). Rather than demonstrating how to build a UI (which is a VB procedure outside the scope of this tutorial) the next section shows you how to customize the report by hard-coding the modifications. With a UI, you would pass your user's selections in place of the hard code.

Write the code that does the modifications

This report is currently based on all regions. We'll modify the selection criteria to include only data from the USA. But first you may need to add a reference to your project.

1. Close the Crystal Report Viewer.
2. From the Project menu select References.
3. Check the Microsoft ActiveX Data Objects Recordset 2.0 Library and click OK.
4. Double-click Form2 from your Project Explorer to open it. This form contains the Report Viewer. Double-click on it to see the code in the main window. It should look something like this:

This code should appear in the General Declarations section.

```
Dim Report As New CrystalReport1
```

This code should appear in the Form_Load event procedure. It tells the viewer what report it should display and then it tells it to display (view) the report.

```
Private Sub Form_Load()
          Screen.MousePointer = vbHourGlass
          CRViewer1.ReportSource = Report
          CRViewer1.ViewReport
          Screen.MousePointer = vbDefault
End Sub
```

This code should appear in the Form_Resize event procedure, it resizes the viewer every time the form is resized.

```
Private Sub Form_Resize()
          CRViewer1.Top = 0
          CRViewer1.Left = 0
          CRViewer1.Height = ScaleHeight
          CRViewer1.Width = ScaleWidth
End Sub
```

5 To add selection criteria to the report. Edit the "Form2" declarations and "Form_Load" procedure to look like this:

```
Dim Report As New CrystalReport1
Dim rs As New ADOR.Recordset
Private Sub Form_Load()
  Screen.MousePointer = vbHourGlass
  rs.Open "Select * from customer where country = 'USA'", _
     "xtreme sample database"
  Report.Database.SetDataSource rs,3,1
  CRViewer1.ReportSource = Report
  CRViewer1.ViewReport
  Screen.MousePointer = vbDefault
End Sub
```

This adds a query to select only the records with "USA" in the "Country" field. The recordset that results from that query will be passed to the report engine and the results displayed.

6 Run the report again to see the changes. Your report should only contain USA data.

7 Now we'll add a text object to the report and manipulate it using code. In the main Report Designer window, right-click just to the right of the graph in the Report Header pane and point to Insert, then select Text Object.

8 Move the text object just above the pie graph. You may have to move the graph down a bit first. Make note of the Name in the Properties window. In this example, the name is "Text5" but the name in your project may be different.

9 Now we'll set the contents of the text object you just added. Open the "Section5" code for editing in the Text Object by double-clicking on the Report Header [Section5] and add some code to the "Section5" "Format" procedure as shown.

```
Private Sub Section5_Format(ByVal pFormattingInfo As Object)
     Text5.SetText "Here is some text from my app"
     Text5.Font.Italic = True
     Text5.Font.Bold = True
     Text5.Font.Size = 14
End Sub
```

10 This will set the text object, change the font to Italic, Bold 14 point and display the results. Run the report again to see the changes.

To complete this exercise, you'll now add some event-driven code to explore how you can make the run time viewer control more interactive.

11 With the Form 2 code window open, use the Object and Procedure list boxes to select the "CRViewer1" object and add the "DrillOnGroup" and "PrintButtonClicked" procedures to the "Form2" code. Edit these new procedures as shown below:

```
Private Sub CRViewer1_DrillOnGroup(GroupNameList As
Variant, ByVal DrillType As
CRVIEWERLibCtl.CRDrillType, UseDefault As Boolean)
     MsgBox "You're drilling down on the " &
     GroupNameList(0) & " group!"
End Sub

Private Sub CRViewer1_PrintButtonClicked(UseDefault As Boolean)
             MsgBox "You clicked the Print button!"
End Sub
```

12 The "DrillOnGroup" procedure is triggered when the user double-clicks on any graph pie slice or report summary field. When this occurs, the text shown will appear in a message box. The "PrintButtonClicked" procedure works similarly when the viewer Print button is clicked. Try both of these procedures to make certain your code works as expected.

Now that you've seen a few simple ways to modify a report, you may want to explore some of the other properties and events in the report engine that you can use in your application. For more sophisticated programming examples, consult the sample applications. For more details on specific methods, properties, or events, use the VB Object Browser or consult the developer’s help files.

RDC Data Access 5

This chapter explains data access through the Report Designer Component (RDC), and describes how to connect to a data source through the Data Explorer. It also provides information on Active Data, the Microsoft Data Environment, and the database drivers used by the RDC.

Overview

In response to the question, “What data can I use when building reports in the RDC?”, the answer is, “You have virtually no limitations.” The RDC has been designed to use virtually any data source with minimal work on your part.

The Report Designer Component supports data access through Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO). Through these three access methods, you can connect to most ISAM, ODBC, and SQL data sources available to Windows applications. In addition, you can create DAO Recordsets, RDO Resultsets, or ADO Recordsets in Visual Basic, then replace the data source used by a report at runtime by calling the SetDataSource method of the RDC runtime Report object.

The Report Designer Component also provides the ability to design reports based on Data Definition files, text files that define the fields, and field types of a database without actually serving as a source of data. Using Data Definition files, you can design a report without depending on the existence of a database at design time, then dynamically create the data at runtime and assign it to the Report object of the Report Designer Component.

If you have installed the full Crystal Reports product, you also have the ability to use the Report Designer Component to connect to any data source that Crystal Reports can connect to. In such a case, the Report Designer Component implements the Crystal Reports user interface that you are already familiar with, so you can quickly connect to any existing data.

Finally, the Report Designer Component also supports Data Environments in Visual Studio 6.0.

Probably the best way to cover all of the data sources is to take a brief look at the Data Explorer dialog box. This dialog box appears when you select Add Database to Report or Set Location, and it lists data source options in a hierarchical tree view.

Visit the Library on the Seagate Software website to get documentation and knowledge base articles about data access.

The Data Explorer

The top three nodes in the Data Explorer were designed to make it more efficient for you to return to favorite data sources, current connections, or data sources you have used in the past. Those nodes are all intuitive, and they are discussed in depth in the *Crystal Reports User's Guide*. We'll be discussing the bottom three nodes in this chapter.

ODBC

The first of these nodes, ODBC, shows a list of ODBC data sources that you have already configured for use. These are local and remote data sources that are not accessed through an object model such as ADO or DAO. The Report Designer Component can use the same database drivers for these data sources that are available through Crystal Reports.

Note: To use these drivers, you must install the full Crystal Reports product.

You can either use an existing data source or create a new one by selecting the **Create New Data Source** option. If you select, Create New Data Source, the RDC displays a Wizard that leads you through the creation process.

You will need to supply:

- the type of data source you want to create
- the driver you want to use
- any driver-specific information that is required.

If you are setting up a machine-independent File Data Source, you will also be asked to supply the name of the file data source you want to save the connection to.

Once you've supplied this information, the Wizard sets up the data source for you.

To connect to an existing ODBC data source

1. Choose the data source.
2. Choose the table(s) that contain the data you want to use.
3. Link the tables (if required) using the Visual Linking Expert.

Once you have done this, the RDC lists the tables you have selected and the fields available in each table under the Database Fields node in the Field Explorer on the left side of the Designer.

Then drag the fields you want onto your report and position them where you want them.

Database Files

This node lists standard PC data sources that reside locally. If your database isn't listed, browse for your file using the **Find Database File** option.

More Data Sources

Use the More Data Sources node to select data sources that can be accessed through OLE DB and native drivers. This includes:

- Microsoft Active Data sources (ADO, DAO, RDO)
- Exchange folders
- Data stored on the local file system
- Mailbox Administration Files from Exchange
- IIS and Proxy server log files
- NT Event logs
- Direct OLE DB connections
- Outlook folders
- Web IIS Log Files

Active Data Sources

When you choose any of the Active Data data sources (ADO, DAO, RDO, or Data Definition Files), the RDC displays the Select Data Source dialog box with the appropriate tools enabled.

ADO

ActiveX Data Objects is the new data connection technology designed to provide a common interface for working with relational databases, mail messaging, event logs, and most any other form of data storage. ADO can be used to connect to any ODBC or OLE DB compliant data source when you design your report in the Report Designer Component.

The resulting ADO Recordset is not directly available from the Report Designer Component, but it can be replaced by alternative Recordset objects. For example, if your Visual Basic application allows users to make choices that can affect the set of data displayed by a report, simply create a new ADO Recordset at runtime, then pass it to the Report Designer Component using the SetDataSource method. The only restriction is that the fields in the new Recordset match the fields originally used to design the report. For more information on using the SetDataSource method, see "SetDataSource method" on page 48.

ADO is the most flexible option for data access. It provides a means of connecting to all kinds of data, including mail messaging, event logs, and Web server site data. In addition, it has been designed, ultimately, to replace DAO and RDO.

- If you are creating new applications using Visual Basic, you should strongly consider ADO for your data connections.
- If you are working with Visual InterDev for developing web sites, you have an additional advantage of being able to use ADO with Active Server Pages.

DAO

Data Access Objects (DAO) is designed primarily for use with local and ISAM (Indexed Sequential Access Method) data sources created through applications such as Microsoft Access, Microsoft FoxPro, and Borland dBASE. Although DAO can be used to connect to ODBC data sources, RDO and ADO provide more powerful options for such data. However, DAO is the oldest of the three technologies, giving it the advantage of being familiar to many Visual Basic programmers. As a result, DAO is also frequently found in existing applications, and applications created with older versions of Visual Basic.

If you are adding the Report Designer Component to a Visual Basic application that already uses DAO, or if you are connecting to a local data source such as an Access or dBASE file, you should consider using DAO to design your reports. Experienced Visual Basic programmers who are familiar with the DAO object model may also want to stick with a known technology. However, if you are working with ODBC or other remote data sources, RDO and ADO may be a better solution.

Once you design your report in the Report Designer Component, information about the connection to your DAO data source is stored with the report, and cannot be accessed directly. However, you can change the data displayed by a report by changing the data source at runtime using the SetDataSource method. A DAO Recordset object may be passed to the report through this method. Keep in mind, though, the Recordset must have fields identical to those in the original data source used at design time.

RDO

Remote Data Objects (RDO) is designed specifically for working with remote data sources. This includes ODBC and most common SQL database systems. In fact, RDO acts as an object-oriented wrapper around the ODBC API. The flexibility of ODBC is available to Visual Basic programmers through the simplicity of a COM based object model. Already a common choice for developers of client/server systems, RDO allows the execution of stored procedures and the processing of asynchronous operations, meaning your users can continue to work while your application processes a data query in the background.

The basic object used to manipulate data in RDO is a Resultset (specifically, the rdoResultset object). A new Resultset can be defined in your Visual Basic application and passed to the Report Designer Component at runtime using the SetDataSource method. The RDO data source used to initially create your report at design time, however, is not available for direct manipulation. Instead, information about the connection to the data source is stored inside the instance of the Report Designer Component that you add to your application. By creating a new Resultset at runtime, though, and passing the Resultset to the Report Designer Component, you can control the data in a report based on user requests and responses.

RDO provides a powerful connection to remote data sources through ODBC. If you are designing a client/ server application, or any application that needs to connect to a large database system such as Microsoft SQL Server, Oracle, or Sybase Adaptive Server, RDO can provide a strong solution. However, RDO limits your application to standard relational database systems. Other sources of data, such as e-mail and messaging servers, system logs, and Internet/intranet server logs are unavailable to RDO. Developers designing web-based applications using Visual InterDev would also be served better by ActiveX Data Objects (ADO).

Crystal Data Object (CDO)

If you develop applications that produce data that does not exist outside of the running application (applications monitoring system or network resources, for example), you can take advantage of the most powerful reporting features in the industry via the Crystal Data Object (CDO). Using CDO, the Active Data Driver, and the Report Design Component, you can create reports that are instant and up to date, without first having to dump the data to a separate database.

The Crystal Data Object is an ActiveX DLL that can be accessed from any Windows development environment that supports ActiveX. By creating a Rowset object, similar to a Recordset, and filling it with fields and data, you design a virtual database table that passes as an ActiveX data source to the Crystal Active Data Driver. Once the CDO Rowset has been created, it can be used just like any other active data source such as DAO or ADO.

CDO, like DAO and ADO, is based on the Component Object Model (COM). Any development environment that supports COM interfaces can dynamically generate a set of data for a report without relying on a database that exists at design time.

Note: The Crystal Data Object does not support Memo or BLOB fields.

Crystal Data Source Type Library

The Crystal Data Source Type Library, like Crystal Data Objects, provides a means for designing customized data sources that can be reported off of using the Active Data Driver. Crystal Data Source, however, unlike CDO, is a type library with an interface that can be implemented in a standard Visual Basic class. Once implemented, the Crystal Data Source interface allows you to manipulate data fully, much like you would manipulate a standard Recordset object in ADO or DAO.

Note: The Crystal Data Source type library is designed for Visual Basic 5.0 or later.

Keep in mind, though, once you add the Crystal Data Source interface to your class, you must implement all methods and properties exposed by the interface.

CDO vs. the Crystal Data Source Type Library

While CDO and the Crystal Data Source Type Library have some similarities, they have been designed as solutions to different data access problems:

- If you need to implement a complete data source in your application that allows runtime movement through records and fields, or if you intend to implement your data source as a separate ActiveX component, consider using the Crystal Data Source Type Library.
- If you need to create a quick and simple means of storing a large amount of data in a convenient package for reporting on, and the data will remain inside the same application as the reporting functionality, then use Crystal Data Objects.

Data Definition Files

A data definition file is a text file that contains information about the kind of data to place in a report instead of information about an actual data source.

- At design time, you can build reports by pointing the Active Data Driver at a data definition file.
- At runtime, point the Active Data Driver to an actual data source and the RDC builds your report from that new data.

While data definition files provide great flexibility in your reporting, and they are still supported in Crystal Reports, they have been superseded with a more elegant technology, Unbound Fields.

Data Environments

Data Environments (introduced with VB 6) allow for interactive creation of hierarchical ADO Objects at design time, and easy access to data at run time. Through the Data Environment Designer you create Data Environment objects to connect with Data sources (Connection Objects), and access data from those sources (via Command Objects) at design time and run time.

You can use a data environment as a data source for reports you create with the Report Designer Component. If your project includes a data environment, the Report Designer Component will allow you to select the data environment as a data source at design time. Runtime reports make full use of the data exposed by a data environment by working with the standard connection, command, or recordset objects provided, much like working directly with an ADO object.

Microsoft Data Link (UDL) files

The Report Designer Component now supports Microsoft Data Link (UDL) files.

Microsoft Data Link Files are comparable to File DSNs in ODBC. UDL files stores connection information to be used by the OLE DB layer. UDL's are available with OLE DB 2.0, which is installed by MDAC 2.x.

To create a UDL file

1. Right-click on the Windows desktop, choose "New" from the popup menu and then choose "Microsoft Data Link". A new UDL file will appear on the desktop. By default, it will be called "New Microsoft Data Link.UDL". You can rename this file.
2. Double-click, or right-click, and choose Properties on this icon to configure the connection. Select the appropriate OLE DB provider in the Provider tab before entering information in the Connection tab. The template for the Connection tab changes based on the OLE DB provider used.

For more information on how to use UDLs, see the Microsoft Data Link help file, "msdasc.hlp".

Report templates using unbound fields

The RDC now enables you to create report templates based on unbound field objects. Then, using minimal code, you can bind those field objects to a data source (or one of several data sources) at runtime. The Report Engine uses the data type and the Name property of the unbound field object to match it with a data field in the recordset.

This technology replaces the need for data definition files. Data definition files are still supported in the Report Designer Component however. For additional information on using Unbound Fields, see "Using fields that bind to data sources only at runtime" in the *Crystal Reports Developers Guide* (*CrystalDevHelp.chm*) or the "Unbound fields" on page 26 and "Pro Athlete Salaries" on page 27 sample applications found in the C:\Program Files\Seagate Software\Crystal Reports\Samples\En\Code\Visual Basic folder.

Database Drivers

For the Report Designer Component to get data to display on the report, it uses a database driver. That driver may be a native driver such as Seagate's native Microsoft Access driver, or it may be an ODBC or an OLEDB driver. Each of these drivers is a separate DLL.

These drivers architecturally work the same way: the Report Designer Component accesses the database for the data it needs.

Crystal Active Data Driver

The one driver that stands out is the Active Data driver (P2smon.dll (32-bit)). The difference with this driver is that it is not intended to be used to connect to your database. Instead, this driver works with recordsets.

Most database applications that you might create use recordsets to send data back and forth between the application and the database. With the Active Data driver, you would pass a populated recordset object, which is sitting in memory, to the Crystal Report Engine. The Report Engine then reads the recordset object to populate the report instead of reading the records in the database itself.

Passing the Recordset

For Crystal Reports to use the recordset in memory, you need to:

- declare the recordset
- run the query
- extract the data.

This recordset can be of type ADO (Data Environment), DAO, RDO, or CDO. The syntax involved in passing a recordset to a report differs depending on the component being used. Refer to the Developer help file for more information.

Note: You must pass a recordset to the report. If you do not pass a recordset, the Report Designer Component will attempt to connect directly to your database.

- If you have an unsecured database such as Microsoft Access, the report will work, but the driver is not being used the way it is intended to be used.
- If the database is secured like a Microsoft SQL Server database, then an error message stating "Server not yet opened" will be displayed because a password was not supplied in order for the report to log on to the database.

SetDataSource method

The SetDataSource method is designed for reports using the Active Data driver. The Method is used to provide information about a data source to the database driver associated with a Database object. For example, if the Crystal Active Data Driver has been used to design the report, this method can be used to provide an active data source for the report, such as a DAO, ADO, or RDO Recordset or a CDO Rowset.

When RDC uses the Active Data Driver

There are three times that the RDC automatically uses the Crystal Active Data Driver:

- When you use the Project button on the Data tab of the Report Expert to select your data source, the RDC will automatically use the Active Data Driver.
- When you choose one of the Active Data options from the More Data Sources node in the Data Explorer, the RDC will also use the Active Data Driver.
- When you use the new methods AddADOCommand or AddOLEDBSource, the RDC will automatically use the Active Data Driver. Both of these methods require you to supply an active ADO connection.

Determining if the RDC is using the Crystal Active Data Driver

If you are having a problem with your data connection and you want to make certain that you are using the Crystal Active Data Driver:

1. Right-click the empty space beneath the field tree in the Field Explorer at the left of the RDC Designer.
2. Select **Set Location** from the shortcut menu when it appears.
3. When the Set Location dialog box appears, look at the Server Type at the bottom of the dialog. If the RDC is using the Active Data Driver, the server type will read Active Data and then give the active data type (For example, ADO, DAO, etc.).

Note: Alternatively, you can select Convert Database Driver from the shortcut report. When you make this selection, the driver currently in use will be grayed out yet visible.

Understanding the RDC Object Model 6

This chapter provides you with an understanding of the Report Designer Component (RDC) Object Model through a high-level overview of how a report is created. The overview is then related to the primary objects and collections contained in the RDC Object Model. More detailed information on a number of the objects, collections, and events is available at the end of the chapter.

Overview

The Report Designer Component is a dual interface object model based on Component Object Model (COM) technology, a standard that allows applications and component objects to communicate with one another. Because the standard doesn't specify how components are structured, but defines only how they communicate with one another other, the RDC can be used in any development environment that supports COM—such as Visual Basic, Visual C/C++, Visual InterDev, etc.

Note: Visit the Library on the Seagate Software website to get documentation and knowledge base articles about the Report Designer Component Object Model.

A high level overview

The RDC is similar in operation to the Form designer in VB: it enables you to create instances of various classes visually.

- You design a form by positioning controls where you want them to appear and then setting their properties.
  - Each of these controls is actually an instance of a class. A command button is an instance of one class. A text box is an instance of another.
- You design a report in the same way, by positioning objects where you want them to appear and then setting their properties.
  - Each of the objects in your report is an instance of a class. A Section is an instance of one class. A text object is an instance of another.

A new report, by default contains five section objects: the Report Header (RH) section, the Page Header (PH) section, the Details section, the Report Footer (RF) section, and the Page Footer (PF) section. The sections are numbered in consecutive order from top to bottom. The top section is one; the bottom section is five. All of these sections together make up the Sections collection. The Sections collection is indexed with a one-based index.

Each of these sections has a single event: Format. The Format event is fired whenever a pass is made through the section as the report is being created. By trapping the format event for a section, you can manipulate the section or any of the objects you place in the section whenever the section is fired.

- Manipulating the section involves actions such as setting its background color or its height.
- Manipulating an object involves actions such as changing a font color in a field object based on a condition being true or changing the report title in response to user input.

You can do just about anything with the report at runtime, with a minimum of code, and all within the VB IDE.

At this very basic level:

- You can place objects in your report.
- You can manipulate the objects as needed in response to various events.

The next level

A report can have more than five sections. You can create these sections yourself if you need additional sections. The RDC will create some for you whenever you group or summarize data.

Sections are numbered consecutively as they are created.

- The first new section created is section 6.
- The second new section is 7, and so forth.

When sections are created, they are added to the Sections collection and indexed appropriately.

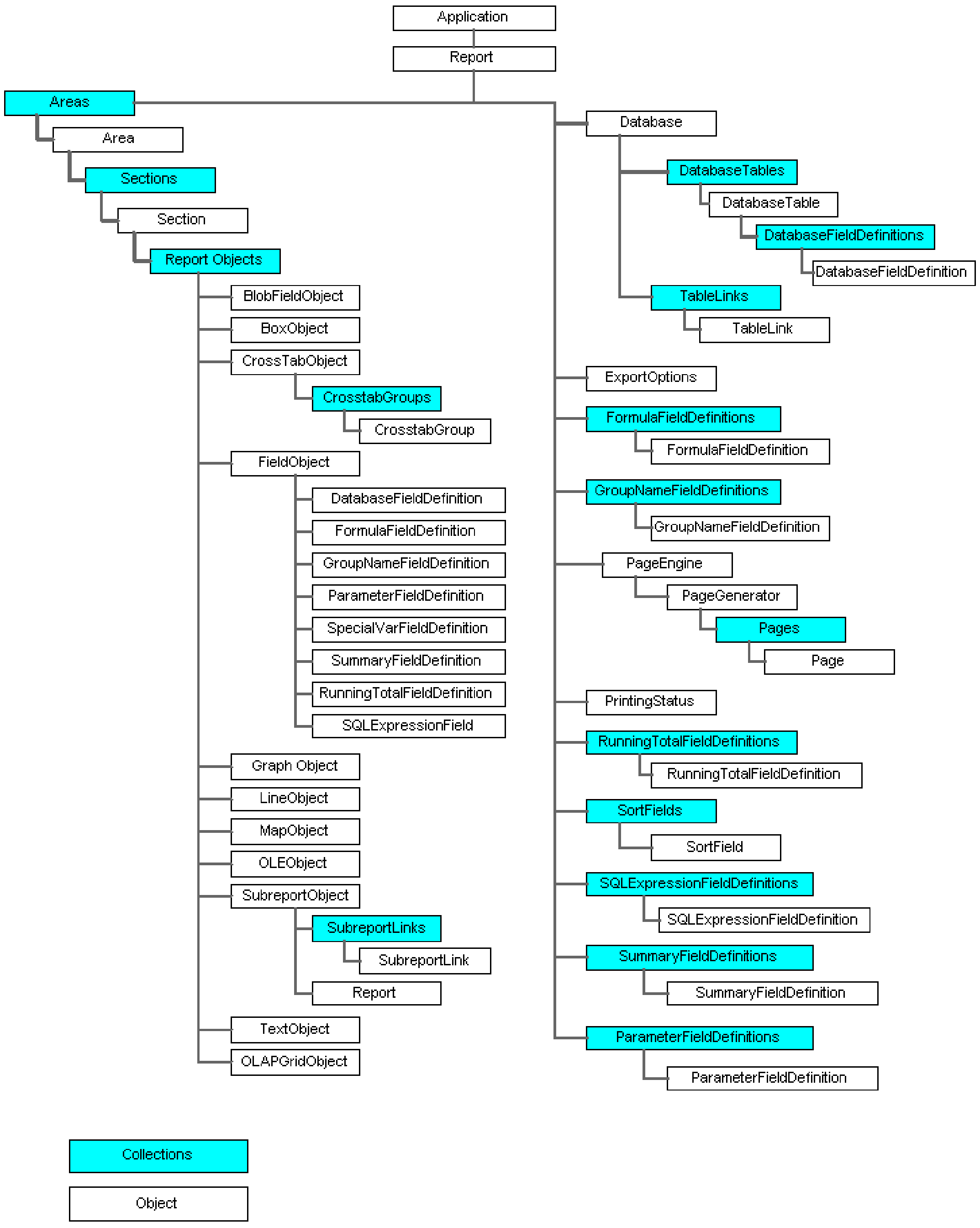
You can place a variety of Report Objects in the sections of your report.

- If they are Field Objects, they could be one of many kinds of fields including database fields, formula fields, parameter fields, group name fields, or many others as well.
- If they are Subreport objects, they can each be viewed as a separate object with its own set of properties. But each subreport is itself a report, and as a report it can contain virtually all of the same kinds of different objects that it's parent report contains.
- If they are Cross-tab objects, they too can be viewed as separate objects or as specialized reports that themselves contain a variety of different objects.

With Subreport objects and Cross-tab objects, you can change the properties of the objects themselves (Subreport Name, Cross-tab Name) and you can change the properties of the objects they contain as well.

The Primary Objects and Collections

The following objects and collections are discussed in the section:


Application
Report
Areas
Area
Sections
Section
Report Objects
BlobFieldObject
BoxObject
CrossTabObject
CrosstabGroups
CrosstabGroup
FieldObject
DatabaseFieldDefinition
FormulaFieldDefinition
GroupNameFieldDefinition
ParameterFieldDefinition
SpecialVarFieldDefinition
SummaryFieldDefinition
RunningTotalFieldDefinition
SQLExpressionField
Graph Object
LineObject
MapObject
OLEObject
SubreportObject
SubreportLinks
SubreportLink
Report
TextObject
OLAPGridObject
Database
DatabaseTables
DatabaseTable
DatabaseFieldDefinitions
DatabaseFieldDefinition
TableLinks
TableLink
ExportOptions
FormulaFieldDefinitions
FormulaFieldDefinition
GroupNameFieldDefinitions
GroupNameFieldDefinition
PageEngine
PageGenerator
Pages
Page
PrintingStatus
RunningTotalFieldDefinitions
RunningTotalFieldDefinition
SortFields
SortField
SQLExpressionFieldDefinitions
SQLExpressionFieldDefinition
SummaryFieldDefinitions
SummaryFieldDefinition
ParameterFieldDefinitions
ParameterFieldDefinition
Collections
Object

The Application object

The top level of the object model is the Application object. Since it is at the top, the RDC always expects it to be there, so there's no need to reference it in your code (generally). There are some specific instances where you must reference it. For more information, see "The Application Object" on page 55.

The Report object

Below the Application object is the Report object. The Report object contains many Collections and objects. For more information, see "The Report Object" on page 56.

The Areas collection

The very top Collection in report is the Areas collection. By default it contains five Area objects. This is a special kind of collection that you may not use too often. We'll talk about that when we discuss the next collection, Sections.

The Sections collection

Every report, including a blank report, has Sections. The Sections are the bands in the report where you place the various report objects. By default, a new report, and thus the Sections collection for that report, contains five sections. Each of the five default sections resides in one of the five default areas—five sections:five areas. When you begin with a blank report, Areas and Sections are essentially the same. So there's little need to reference an area unless you have a specific need to do so.

For example, you want to create several of one kind of section. In this example, you want to create several Details sections. In this case, the five areas:five sections relationship is broken. To refer to those Details sections as a unit, you can think of one Details Area now containing several Details sections. When you reference the area, you are referencing all of the sections in that area.

Thus, unless you have a special need, you can think of Sections as having Report for a parent. If you have the need, or if you want to be very specific, you can also think of Sections having Area for a parent. However you think of this part of the object model is fine. As long as you have five sections and five areas, you can access Sections through Areas, or you can go directly to Sections. Your code will work either way. For more information, see "The Sections Collection" on page 59.

The ReportObjects collection

ReportObjects is a collection that can contain many things. With the exception of Areas and Sections, everything else in a report is a report object including Database field objects, Parameter field objects, and Text objects. Note that we used a lower case "r" and "o" for our spelling of "report object". The reason is there is no "ReportObject" in the ReportObjects collection. The ReportObjects collection is just a convenient way to think about these objects and it is useful in your code.

Now many of these objects that are considered to be members of ReportObjects are also members of more specific collections.

- A SortField object, for example, is a member of the ReportObjects collection. It is also a member of the SortFields collection.
- A FormulaFieldDefinition object, likewise, is a member of the FormulaFieldDefinitions collection.

You can access an object through the ReportObjects collection or through its namesake collection. This provides you with much flexibility in your coding. See "The ReportObjects Collection" on page 58.

The FieldDefinitions collection

In the ReportObjects collection, many of the members have the word "field" in their names: GroupNameFieldDefinition, ParameterFieldDefinition. Objects that contain the word "field" are also a part of the FieldDefinitions collection. As was the case with ReportObjects, there is no such thing as a generic FieldDefinition in the FieldDefinitions collection. Instead there are specific kinds of fields including FormulaFieldDefinition, SpecialVarFieldDefinition, SummaryFieldDefinition, and others as well.

Following this up the hierarchy, we see that a ParameterFieldDefinition is a member of:

- its namesake collection, ParameterFieldDefinitions
- the FieldDefinitions collection
- the ReportObjects collection.

The Subreport Object

The Subreport Object is part of the ReportObjects collection. A subreport is a report within a report; it is a complete report with its own Areas, Sections, and ReportObjects. Once you access a Subreport object, you manipulate the objects in the subreport the same way you would with objects in the main report. For more information, see "The Subreport Object" on page 57.

The Database Object

The Database object represents a data source for the report. But a data source doesn't mean all the data available. It means only a subset of data that is useful for the report. That subset can be defined as collections of tables and, within each of those tables, collections of fields. When you think about a data source in this way, the data side of the RDC object model becomes clear. For more information, see "The Database and Database Tables Objects" on page 56.

The DatabaseTables collection

The Database object contains a DatabaseTables collection. This collection contains a number of DatabaseTable objects which comprise all the Tables which contain data used in the report.

The DatabaseFieldDefinitions collection

Each DatabaseTable object contains a DatabaseFieldDefinitions collection which holds all of the fields from the table that are available for the report. Each of those fields is called a DatabaseFieldDefinition.

Object Considerations

The Application Object

Not all development environments support ActiveX designers. Visual C++ and Delphi, for example, are two environments where you can use the functionality of the RDC automation server but you can't add the RDC directly to your projects. In these cases, you must access the Report Designer's Report object by creating an instance of the Application object and then calling the OpenReport method as well.

In other situations, you may need to use separate report files in your application that were created or edited using Crystal Reports. An advantage of using such standalone report files is that through Crystal Reports, you can save report data with the report file, eliminating the need of maintaining a connection to a data source at runtime. In this case you will need to create an instance of the Application object and then call the OpenReport method as well.

The following code sample demonstrates the process of obtaining an Application object and opening a report file in Visual Basic:

```
Dim CRXApplication As CRAXDRT.Application
Dim CRXReport As CRAXDRT.Report
    Private Sub Form_Load()
       Set CRXApplication = CreateObject("CrystalRuntime.Application")
       Set CRXReport = CRXApplication.OpenReport("C:\Reports\Sales.rpt")
    End Sub
```

When this code finishes, rpt is a valid Report object and can be used just like any Report object you would obtain through the Report Designer Component.

Note: The sample call to CreateObject above uses a version independent Prog Id for the Report Designer Component. The correct Prog Id for this version of the

Report Designer Component is CrystalRuntime.Application.8, but the version independent Prog Id should use the most recent version of the component installed on your system.

The Report Object

Many existing ActiveX object models require declaring and creating a high level object, such as an Application object, before you can work directly with the data. The RDC, however, allows direct access to the Report object, giving you control over your reports with a small amount of Visual Basic code.

Assuming you have added the RDC to a Visual Basic application, and the (Name) property of that component is set to CrystalReport1, you can use code similar to the following to obtain a Report object representing that component:

Example:

```
Dim cr As CRAXDRT.Report
Set cr = New CrystalReport1
```

- The first line declares cr as Report object from the CRAXDRT object library, the Report Designer Component's object library.
- The second line of code defines cr as a new instance of the report in the CrystalReport1 component in your application.

The Field Object

All relational databases have two standard features: records and fields. Records contain the actual data, while fields define what type of data the records contain.

- You can control the records through Recordset and Resultset objects exposed by ADO, RDO, and DAO.
- You can manipulate the fields in the data source through these objects as well.

To manipulate the fields that appear in your report, however, you must either:

- know the name of each database field you want to manipulate and reference each field directly, or
- cycle through the fields in the ReportObjects collection to identify those fields that you want to manipulate.

For examples of how to use either of these methods, see "Referencing objects in a report" on page 66.

The Database and Database Tables Objects

All reports must connect to a data source to obtain data. The most commonly used data source is a relational database. The Report Designer Object Model, therefore, has provided objects, properties, and methods specific to working with databases.

The Database object is available directly from the Report object and represents the data source used by the report. This data source can be changed using the Database object's SetDataSource method. In addition, the Database object provides the Tables property, a read-only property that gets the DatabaseTables collection of DatabaseTable objects. You have access to log on information (for password protected systems), database driver names, and database locations through a DatabaseTable object. Use code similar to the following in order to work with the Database and DatabaseTable objects:

```
Dim Report As New CrystalReport1
Dim tableName As String
Dim dbTable As CRAXDRT.DatabaseTable
Report.Database.Verify() ‘ Verifies a valid connection to the database
For Each dbTable In Report.Database.Tables
   tableName = dbTable.Name
Next
```

The Subreport Object

A SubreportObject object is another report inside the original report. Once you have obtained a SubreportObject, you can work with any aspect of it just as if it were a standard Report object.

You can obtain a SubreportObject through the ReportObjects collection. The following example shows you how to iterate through the sections of a report and change the background color of each subreport to magenta.

```
Dim Report As New CrystalReport1
Dim subReport As SubreportObject
Dim sect As Section
Dim rptObject As Object

For Each sect In Report.Sections
    For Each rptObject In sect.ReportObjects
        If rptObject.Kind = crSubreportObject Then
            Set subReport = rptObject
            subReport.BackColor = RGB(255, 0, 255)
            Set subReport = Nothing
        End If
    Next
Next
```

Note: Currently, the Crystal Report Designer Component does not support subreports inside of subreports. The report iterations cannot go more than one subreport deep. However, you can have multiple subreports inside the main report.

The CrossTabObject

A CrossTabObject object in the Report Designer represents a single cross-tab in your report. Cross-tabs are specialized subreports. Even if you design your primary report as a cross-tab, it is added to the report page as a separate object inside the report.

You can obtain CrossTabObject objects from a report much like you obtain subreports. Since a CrossTabObject is a report object like a field or a subreport, you can access it through the ReportObjects collection.

CrossTabObjects can be manipulated in the same way as other report objects. However, due to the complexity of this object, few properties and methods are exposed.

The following code searches for cross-tabs in a report and applies formatting features to make them stand out from the rest of the report.

```
Dim Report As New CrystalReport1
Dim xtObject As CrossTabObject
Dim sect As Section
Dim rptObject As Object
For Each sect In Report.Sections
             For Each rptObject In sect.ReportObjects
                 If rptObject.Kind = crCrossTabObject Then
                     Set xtObject = rptObject
                 xtObject.BorderColor = RGB(255, 0, 0)
                     xtObject.HasDropShadow = True
                     Set xtObject = Nothing
                 End If
             Next
Next
```

Collection Considerations

The ReportObjects Collection

The ReportObjects collection of a report Section object contains all report objects in that section. Report objects may be Text objects, Fields, Subreport objects, or Cross-tabs. To be able to work with a particular report object, you must first obtain the object, and then determine what type of object it is.

Note: If you want to work with all report objects in a section, consult IReportObject in the Object Browser for a list of properties that are common to all report objects.

Usually, report objects can be addressed directly in your code based on their Name property.

However, if you intend to work with several objects in a section, you need to refer to them through a Section object in the report. The following code locates the last object in the last section of the report and assigns a value to a String variable based on the type of object it is.

```
Dim Report As New CrystalReport1
Dim sect As Section
    Dim rptObject As Object
    Dim objKind As String
    Dim lastSectWithObject As Integer
lastSectWithObject = Report.Sections.Count
Set sect = Report.Sections.Item(lastSectWithObject)
Do While sect.ReportObjects.Count <= 0
            lastSectWithObject = lastSectWithObject - 1
    Set sect = Report.Sections.Item(lastSectWithObject)
Loop
Set rptObject = sect.ReportObjects.Item(sect.ReportObjects.Count)
Select Case rptObject.Kind
            Case crBlobFieldObject
            objKind = “BLOB field object”
            Case crBoxObject
            objKind = “Box object”
            Case crCrossTabObject
            objKind = “CrossTab object”
            Case crFieldObject
            objKind = “Field object”
            Case crGraphObject
            objKind = “Graph object”
            Case crLineObject
            objKind = “Line object”
            Case crOleObject
            objKind = “OLE object”
            Case crSubreportObject
            objKind = “Subreport object”
            Case crTextObject
            objKind = “Text object”
            Case Else
            objKind = “Unknown object”
End Select
```

Note: Instead of using the Item property as shown, you can reference a report object directly using its index, for example, ReportOptions(1). You can also reference it directly using its name property by including the report object name in quotes, for example, sect.ReportObjects("Field5") instead of sect.ReportObjects(1).

The Sections Collection

All report objects, such as Fields and Text objects, are contained within Sections. Often, you can obtain these directly by referring to the name property of an object in your code.

Note: There may be times, however, when you need to obtain an object through the report section it is in. At other times, you may have a need to make changes to the section itself.

Every report contains a collection of its sections, stored in the Sections property of the report object. You can access individual sections through this collection. For example, the code below sets the height of the first section of the report (the Report Header) to half an inch (720 twips) and suppresses the second section of the report (the Page Header) so that it will not appear.

```
Dim Report As New CrystalReport1
Report.Sections.Item(1).Height = 720
Report.Sections.Item(2).Suppress = True
```

For information on how to obtain and work with other objects within a Section object, such as fields and text objects, refer to "The ReportObjects collection" on page 53.

Each section in a report is itself a collection. The collection contains all of the report objects in that section. When designing your application, be aware that when a section is being formatted, all objects in that section are also being formatted. Also, all other sections and objects outside of the current section are not being formatted. This information can affect how data is displayed in various sections of the report, depending on your code.

Event Considerations

The RDC directly supports four events:

- AfterFormatPage
- BeforeFormatPage
- FieldMapping
- NoData.

The RDC supports the Format event for the Section object as well. These new events are discussed in detail in the Developer's online help.

The Format event for the Section object

The Crystal Report Designer Component provides a Format event for every section of your report. This allows you to control report formatting and output dynamically at runtime. For example, you can apply conditional formatting during the Format event, based on conditions that exist only at runtime.

The Format event is a standard Visual Basic event that you can program by displaying the Code View of the Report Designer Component. Assuming the Report Designer has been named CrystalReport1 in your Visual Basic project:

1. In the Visual Basic Project window, select the CrystalReport1 designer from the Designers folder.
2. Click the View Code button in the toolbar at the top of the Project window. A Code window will appear for the CrystalReport1 designer component.

3 In the object drop-down list box at the top left of the Code window, select a Section object that you want to apply code to. Notice that Section objects are numbered in the order that they appear on your report, from the top of the Report Design window to the bottom. For instance, if you select the Section1 object, your code will apply to the Report Header section of your report. These Section objects are labeled for you at the top of each section in the Report Designer window.

Notice that when you select a Section object, the Format event is automatically selected for you in the event drop-down list box at the top right of the Code window. Format is the only event available to the Section object.

When writing code for the Format event, keep in mind that not all properties and methods for all objects are available during the Format event. Many properties are available on a read-only basis. If you are not sure about a property or method, refer to the specific property or method name in the Object Browser or the Report Designer Object Model reference section of the Developer help file.

The Format event receives a single argument from the Report Designer Component. The pFormattingInfo argument is an object of type FormattingInfo. The FormattingInfo object has only three properties:

- IsEndOfGroup - This property is true if the section being formatted is a Group Footer.
- IsRepeatedGroupHeader - This property is true if the section being formatted is a repeated Group Header. Repeated Group Headers appear when a group extends to two or more pages, and the group has been formatted to repeat information that appears in the Group Header on the second, third, etc. pages. This property is only true if the Group Header is the second, third, etc. instance of the Group Header. It is false for the original first instance of the Group Header.
- IsStartOfGroup - This property is true if the section being formatted is a Group Header.

Note: When designing your application, be aware that when a section is being formatted, all objects in that section are also being formatted. Also, all other sections and objects outside of the current section are not being formatted. This information can affect how data is displayed in various sections of the report, depending on your code.

Note: If you are using the Format event to calculate values and you need to carry that value across sections (as in a running total, for example), you will need to use a Report Variable to store your ongoing total. Report Variables are new to Crystal Reports. For further information on Report Variables, please see ”Report Variables” in the *Crystal Report Developers Help* (*CrystalDevHelp.chm*).

The Visual Basic Initialize and Terminate events

In addition to the report-specific events, the Report object can also produce the standard Visual Basic Initialize and Terminate events.

The Initialize event is fired when you first reference the Report object at runtime. For example, your application may contain a global variable that represents the Report object of the Report Designer that you added to your application at design time:

```
Dim Report As New CrystalReport1
```

In this case, declaring and setting the Report variable fires the Initialize event. The Terminate event will be fired when this variable is set to Nothing:

```
Set Report = Nothing
```

The Initialize event and the Terminate event are fired only once for each Report instance. With that in mind, you can make many changes to your report within the event procedure code for each of these events:

```
Private Sub Report_Initialize()
   ‘ Add report initialization code here
End Sub
Private Sub Report_Terminate()
   ‘ Add report clean up code here
End Sub
```

- Use the Initialize event to make broad changes that affect the entire report. For instance, you could assign a new data source using the SetDataSource method.
- Use the Terminate event to clean-up of any variables or objects that you created during the Initialize event. If, for instance, you created a new ADO Recordset object in the Initialize event, you can use the Terminate event to set this Recordset object equal to Nothing, freeing up system memory and resources.

RDC Programming 7

This chapter provides solutions to a number of common runtime challenges. Most of these challenges involve accessing an object and then changing the properties. The chapter first discusses special considerations when programming with the Report Designer Component (RDC); it then presents the two general methods to access an object in a report or subreport. The examples assume that you have used one of these methods to locate the object of interest in order to concentrate on the special considerations that you may have with each of the various procedures.

Special Considerations

Object naming considerations

If your project includes other libraries that contain objects with names identical to those found in the Crystal Report Engine Object Library, you will encounter conflicts unless you reference the objects using a prefix that identifies their source. For example, if you have included the DAO Library with the RDC runtime object library in your project, you will find that both libraries include a Database Object. In order to avoid conflicts, you must prefix the objects as follows:

```
CRAXDRT.Database
```

for the object from the RDC runtime Object Library, or

```
DAO.Database
```

for the object from the DAO Library.

Scope considerations

When you specify a data source using code, it is important you place the code in the appropriate location in your project.Otherwise, you will face “no data” or runtime error situations because the report may be searching for data that has gone out of scope.

By default, the program does not read data into the report until you actually run the report (print it, preview it, etc.). Anything that terminates the connection to the data source before this time will cause out of scope problems.

Typical out-of-scope problems include:

- Declaring a local object variable for the recordset or resultset in the Form_Load event procedure. While this may seem appropriate, the variable, and thus the data it references, will go out of scope as soon as the form stops loading.
- Similarly, declaring a local object variable in the Report_Initialize event procedure will go out of scope because the recordset is discarded at the end of the procedure. That procedure ends before any records are read from the recordset.

Both problems can be corrected by declaring the variable for the recordset at the module level.

- A third problem can occur when you declare a global variable for the report and try to populate it with a recordset that only has procedural scope. The following code provides a good example of this:

```
Dim CRXReport As New CrystalReport1
Private Sub Form1_Load()
   Call SetData
       CRViewer1.ReportSource = CRXReport
       CRViewer1.ViewReport
    End Sub
Private Sub SetData()
   Dim rs As New ADOR.Recordset
       rs.Open “Select * From Customer”, “Xtreme Sample Database”
     CRXReport.Database.SetDataSource rs, 3, 1
    End Sub
```

In this example, although the CRXReport object is global (for the Form), the rs object exists only for the life of the SetData procedure. Even though it is assigned to the Report object before SetData finishes, the object, and the data that rs represents goes out of scope and is invalid for the Load event. Thus, the code will fail.

This problem can be solved using the ReadRecords method. ReadRecords forces the report to read the records into the report so that they are internal to the Report object itself rather than remaining as a separate Recordset object referenced by the report.

```
Private Sub SetData()
   Dim rs As New ADOR.Recordset
       rs.Open “Select * From Customer”, “Xtreme Sample Database”
     CRXReport.Database.SetDataSource rs, 3, 1
     CRXReport.ReadRecords
    End Sub
```

In this case, the records are read into the report prior to the end of the procedure. Thus, when the procedure ends, the recordset object may be discarded but the records are still available for use.

Index considerations

The various collections in the RDC object model are all one-based, that is, the first member of a collection has index number 1, the second member has index number 2, and so forth. Collection members are indexed consecutively in the order that they are created.

Dual Interface considerations

The Report Design Component provides dual interfaces. As a result, development environments such as Visual Basic and Visual C++ can bind with vTable interfaces at compile time, and less robust environments, VBA, ASP, for example, can use the IDispatch interface for runtime-only binding.

It is important that you use the more efficient vTable interface if it is supported by your development environment.

- When you use vTable binding, you have “early binding” which means the VB compiler can identify syntax errors at compile time when they’re easier to identify and fix than when they appear only at runtime.
- It enables the VB IDE to speed your coding process by providing hints as you code through its “Auto List Members” and “Auto Quick Info” capabilities.

When you work with Visual Basic, always use the more efficient vTable interface as long as you Dim your object variables using specific class names instead of using the more generic “Object” type.

Two methods to access an object in a report or subreport

Referencing objects in a report

Before you work with a report object, you have to reference it (open it). You can reference a report object either explicitly or implicitly:

- Referencing the object Explicitly is the easiest and quickest method for referencing a report object. When you use this method, your code can reference the report and its objects directly.
- Referencing a report object Implicitly is a more generic approach. This approach allows your code to reference the report and its objects indirectly.
- You can access objects in a report either explicitly or implicitly.

Explicit reference – an object in the report

Assume that in your report you have a field with the Name property “OrderID”. Since you know the field name, you can use code similar to this to reference the OrderID field directly.

```
Dim CRXReport As New CrystalReport1
Dim CRXReportField As FieldObject
Set CRXReportField = CRXReport.OrderID
```

Note: Now that you have the OrderID field, you can put your code here to work with the properties of that field.

Implicit reference – cycling through the sections

If you are planning to work with a number of different kinds of report objects, you can cycle through the sections, test each of the report objects for kind, and manipulate them as you wish. You can do that using code similar to this:

```
    Dim CRXReport As New CrystalReport1
    Dim CRXTables As CRAXDRT.DatabaseTables
    Dim CRXTable As CRAXDRT.DatabaseTable
    Dim CRXSections As CRAXDRT.Sections
    Dim CRXSection As CRAXDRT.Section
    Dim CRXSubreportObj As CRAXDRT.SubreportObject
    Dim CRXReportObjects As CRAXDRT.ReportObjects
    Dim CRXReportObject As Object
```

Note: You put this code in the Form_Load event of the startup form which contains the Report Viewer.

Start by getting the sections from the Main report.

```
Set CRXSections = CRXReport.Sections
```

You begin cycling through each section in the main report.

```
For Each CRXSection In CRXSections
```

In each section, you get all the objects in the section.

```
Set CRXReportObjects = CRXSection.ReportObjects
```

You cycle through the objects.

```
For Each CRXReportObject In CRXReportObjects
```

You test the objects to see if they're subreports.

```
If CRXReportObject.Kind = crSubreportObject Then
```

When you find a subreport, you get a hold of it.

```
Set CRXSubreportObj = CRXReportObject
```

Before you work with an object in a report, you have to reference it (open it).

Implicit reference – through a collection

If you are planning to work with a specific kind of report object, you can access the object by referencing the collection that contains that object. You reference a collection in a report through the appropriate Report property. Those properties, and the collections they get, are as follow:

Property	**Gets**
Areas	Areas collection (Areas)
Database.DatabaseTables	Database Tables collection
FormulaFields	Formula Fields collection (FormulaFieldDefinitions)
GroupNameFields	Group Name Fields collection (GroupNameFieldDefinitions)
RunningTotalFields	Running Total Fields collection (RunningTotalFieldDefinitions)
GroupSortFields	Group Sort Fields collection (SortFields)
ParameterFields	Parameter Fields collection (ParameterFieldDefinitions)
RecordSortFields	Record Sort Fields collection (SortFields)
Sections	Section collection (Sections)
SQLExpressionFields	SQL Expression Fields collection (SQLExpressionFieldDefinitions)
SummaryFields	Summary Fields collection (SummaryFieldDefinitions)

Once you have the collection, you can cycle through the collection to locate the appropriate object you want. The following example shows you how to access the FormulaFieldDefinitions collection:

You can put this code in the General Declarations.

```
Dim CRXReport As New CrystalReport1
Dim CRXFormulaField As CRAXDRT.FormulaFieldDefinition
```

You can put this code in the Form_Load event procedure.

```
Private Sub Form_Load()
```

This code gets the collection.

```
Set CRXFormulaFields = CRXReport.FormulaFields
'You can put your code here to cycle through the collection to get the
specific object you want
```

Finally you can view the report.

```
Crviewer1.ReportSource = CRXReport
Crviewer1.ViewReport
End Sub
```

Referencing objects in a subreport

Before you work with a subreport, you have to reference it (open it). Similar to referencing objects in a report, you can do this either explicitly or implicitly.

Explicit reference – the subreport itself

Assume that in your report you have a subreport with the Name property "Orders". Since you know the subreport name, you can use code similar to this to reference that subreport directly:

```
Dim CRXReport As New CrystalReport1
Dim CRXSubReport As SubreportObject
Set CRXSubReport = CRXReport.Orders
```

Note: Now that you have the subreport, you can add code to manipulate the subreport.

Explicit reference – an object in the subreport

Assume that in your report you have a subreport with the Name property "Orders" and that, in that subreport, you have a field with the Name property "OrderID". Since you know the subreport and field names, you can use code similar to this to reference the OrderID field directly:

```
Dim CRXReport As New CrystalReport1
Dim CRXSubReport As SubreportObject
Set CRXSubReport = CRXReport.Subreport1
Dim CRXSubReportField As FieldObject
Set CRXSubReportField = CRXReport.Orders_OrderID
```

Now that you have the OrderID field, you can add code to work with the properties of that field.

Please note the special way you reference a field in a subreport by name. You must use:

- the name of the subreport
- an underscore
- the name of the field.

The result, in this example, Orders_OrderID, appears to be a field in myReport with the name Orders_OrderID. It is actually a field in the Orders subreport; the program uses dynamic type info to generate these names at runtime. You can find the names by looking at the Project in the Object Browser.

Implicit reference

Though this approach is more complex, it is not bound to a particular report, and the code can be reused with minor modifications for other reports that contain subreports. You can use code similar to the following to access any and all subreport(s) in a particular report.

You can declare the Variables in the General Declarations section of a form.

```
Dim CRXReport As New CrystalReport1
Dim CRXTables As CRAXDRT.DatabaseTables
Dim CRXTable As CRAXDRT.DatabaseTable
Dim CRXSections As CRAXDRT.Sections
Dim CRXSection As CRAXDRT.Section
Dim CRXSubreportObj As CRAXDRT.SubreportObject
Dim CRXReportObjects As CRAXDRT.ReportObjects
Dim CRXSubreport As CRAXDRT.Report
Dim CRXReportObject As Object
```

You put this code in the Form_Load event of the startup form which contains the Report Viewer.

You start by getting the sections from the Main report.

```
Set CRXSections = CRXReport.Sections
```

You begin by cycling through each section in the main report.

```
For Each CRXSection In CRXSections
```

In each section, you get all the objects in the section.

```
Set CRXReportObjects = CRXSection.ReportObjects
```

You cycle through the objects.

```
For Each CRXReportObject In CRXReportObjects
```

You test the objects to see if they’re subreports.

```
If CRXReportObject.Kind = crSubreportObject Then
```

When you find a subreport, you get a hold of it.

```
Set CRXSubreportObj = CRXReportObject
```

Finally, you open the subreport and treat it as you would any other report.

```
Set CRXSubreport = CRXSubreportObj.OpenSubreport
```

Note: This is where you put the code for manipulating the subreport.

```
End If
Next CRXReportObject
Next CRXSection
```

Runtime examples

The following topics are discussed in this section:

Working with Cross-Tabs

Modifying a cross-tab at runtime

A CrossTabObject object in the Report Designer represents a single cross-tab in your report. Cross-tabs are specialized subreports. Even if you design your primary report as a cross-tab, it is added to the report page as a separate object inside the report.

CrossTabObject objects can be obtained from a report much like subreports. A CrossTabObject is implemented as a single report object accessible through the ReportObjects collection.

The following code searches for cross-tabs in a report and applies formatting features to make them stand out from the rest of the report.

```
Dim CRXReport As New CrystalReport1
Dim CRXXtObject As CRAXDRT.CrossTabObject
Dim CRXSection As Section
Dim CRXReportObject As Object
For Each CRXSection In Report.Sections
    For Each CRXReportObject In sect.ReportObjects
        If CRXReportObject.Kind = crCrosstabObject Then
            Set CRXXtObject = CRXReportObject
            CRXXtObject.BorderColor = RGB(255, 0, 0)
            CRXXtObject.HasDropShadow = True
            Set CRXXtObject = Nothing
                End If
            Next
Next
```

Although cross-tabs are much like subreports, because of their specialized handling of data, there are fewer properties available to the CrossTabObject object than to the SubreportObject object. Before trying to use a property with a cross-tab in your report, verify that the property is available to the CrossTabObject object.

Working with Data sources

Changing an ADO data source location – new methods

In Crystal Reports 8, there are two new methods to add ADO data sources to a report:

- AddADOCommand takes as parameters an active ADO connection and an ADO command object. In this example, a new connection is created to a database, and the resulting recordset is assigned at runtime to the report.
- AddOLEDBSource takes as parameters the name of an active ADO connection and the name of a table that can be accessed through that connection.

In this example, a connection set up through the VB Data Environment is used as a data source. Both methods can be used with either the VB Data Environment or another data source created at runtime.

```
Option Explicit
Dim CRXReport As New CrystalReport1
```

The ADO connection to the local database.

```
Dim cnn1 As ADODB.Connection
Dim datcmd1 As ADODB.Command
```

Demonstrate the use of AddADOCommand by opening an ADO data command and adding the data source to the report.

```
Private Sub cmdADO_Click()
    Dim strCnn As String
```

Open the data connection.

```
    Set cnn1 = New ADODB.Connection
    strCnn = "Provider=MSDASQL;Persist Security Info=False;Data
Source=Xtreme Sample Database;Mode=Read"
    cnn1.Open strCnn
```

Create a new instance of an ADO command object.

```
    Set datcmd1 = New ADODB.Command
    Set datcmd1.ActiveConnection = cnn1
    datcmd1.CommandText = "Customer"
    datcmd1.CommandType = adCmdTable
```

Add the datasource to the report.

```
    CRXReport.Database.AddADOCommand cnn1, datcmd1
End Sub
```

Adding a data source and a field using AddOLEDBSource

This section demonstrates the use of AddOLEDBSource. One line of code:

- opens an ADO data source
- adds the data source to the report, and then
- adds a field to the report that uses that data source.

In this example, we are using an OLEDB source created in a VB Data Environment.

```
CRXReport.Database.AddOLEDBSource DataEnvironment1.Connection1,"Customer"
```

Setting the data source for a subreport

This section describes how to change the database location for different types of data sources

- Native
- ODBC
- Active Data.

You change the database location (change the data source) for a subreport in much the same way as you set the database location for the main report. Before you can change the database location for a subreport, you first need to open the subreport. For more information, see "Referencing objects in a subreport" on page 68.

Once you have opened the subreport you can change the database location. The following examples show you how to do that for a variety of different types of data sources.

Native connection to PC database

If the datasource of the subreport is a PC-type database (for example, MS Access), and the report is connecting natively (P2BDAO.DLL), then you can use code similar to the following:

1 Get the Tables collection for the subreport.

```
Set CRXTables = CRXSubreport.Database.Tables
```

2 Get the first table from the Tables collection.

```
Set CRXTable = CRXTables.Item(1)
```

3 Finally, set the location of the .mdb file.

```
CRXTable.Location = "C:\My_Application\My_DB.mdb"
```

Native connection to SQL database

If the datasource of the subreport is of an SQL-type (for example, MS SQL Server), and the report is connecting natively (P2SSQL.DLL), then you can use code similar to the following:

1 Get the Tables collection for the subreport.

```
Set CRXTables = CRXSubreport.Database.Tables
```

2 Get the first table from the Tables collection.

```
Set CRXTable = CRXTables.Item(1)
```

3 Finally, set the location of the .mdb file.

```
CRXTable.SetLogOnInfo <servername>, <databasename>, <userid>, <password>
```

ODBC connection to a PC or SQL database

If the datasource of the subreport is a PC-type (for example, MS Access) or SQL-type (for example, MS SQL Server), and the report is connecting via ODBC (P2SODBC.DLL), you can use code similar to the following:

1 Get the Tables collection for the subreport.

```
Set CRXTables = CRXSubreport.Database.Tables
```

2 Get the first table from the Tables collection.

```
Set CRXTable = CRXTables.Item(1)
```

3 Finally, set the location of the .mdb file.

```
CRXTable.SetLogOnInfo <ODBC_DSN>, <databasename>, <userid>, <password>
```

Active data connection to a PC or SQL database

If the datasource of the subreport is an active data source (ex. DAO, RDO, ADO, CDO, etc.), and the report is connecting using the active data driver (P2SMON.DLL), then you can use code similar to the following:

1 Get the Tables collection for the subreport.

```
Set CRXTables = CRXSubreport.Database.Tables
```

2 Get the first table from the Tables collection.

```
Set CRXTable = CRXTables.Item(1)
```

3 Finally, set the location of the.mdb file.

```
CRXTable.SetDataSource rs, 3
```

There are many ways that you can access a subreport, and change the database location for a subreport. The above examples are simplified and generic so that they can be used and altered to accommodate an application with any report.

Connecting to OLEDB providers

Each OLEDB provider requires a specific connection string. The following sample connection strings are for use in the "Connection" input box of the "ADO and OLE DB" option on the "Select Data Source" dialog box for the Active Data report expert:

Microsoft Jet 3.51 OLE DB Provider

Provider=Microsoft.Jet.OLEDB.3.51;Data Source=c:\Access\Northwind.mdb

Note: At this time, it is not possible to use secured Access databases in the Crystal Report expert.

- This OLE DB provider appears to only work with 32-bit Access databases.
- Notice that the database table and query names do not appear in the drop-down boxes. This has been tracked. The option is to use the SQL textbox instead.

Microsoft OLEDB Provider for SQL Server

Syntax:

```
Provider=SQLOLEDB;SERVER=SName;DATABASE=DBName; UID=MyID; PWD=MyPWD
```

Example:

```
Provider=SQLOLEDB;Server=TechTest;Database=Pubs;UID=Admin;PWD=sa
```

Microsoft OLEDB Provider for ODBC Drivers

Syntax:

```
Provider=MSDASQL;DSN=MyDSN;UID=MyUID;PWD=MyPassword
```

Examples:

For secured Data Sources:

```
Provider=MSDASQL;DSN=Xtreme Sample Database;UID=Admin;PWD=Password
```

For non-secured data sources type the DSN:

```
Xtreme Sample Database
```

Note: If it is OLE DB that is being used as the provider for ODBC, select the first option of the "Select Data Source" dialog box ("ODBC(ADO)") which uses that specific provider by default.

Microsoft OLEDB Provider for Oracle

Syntax:

```
Provider=MSDAORA;Password=;User ID=;Data Source=
```

Example:

```
Provider=MSDAORA;Password=MyPassword;User ID=Admin;Data
Source=MyOracleServer
```

Note: It will probably be necessary to re-type the UserID and Password in the corresponding text boxes of the Select Recordset dialog box.

Connecting to a secure Access session

If your reports connect to secure Microsoft Access sessions, you must provide session information at runtime using the SetSessionInfo method of the DatabaseTable object. This method accepts a user ID and password that are passed to the Access session to provide a connection to the database.

Session information must be added to your code using the SetSessionInfo method, even if you specified session information at design time. Session information specified at design time is not stored with the Report Designer Component.

Assuming all tables in your report are from the same secured Access database, you can use code similar to the following to set the Access session information:

```
Dim CRXReport As New CrystalReport1
Dim CRXTable as CRAXDRT.DatabaseTable
For Each CRXTable In CRXReport.Database.Tables
    CRXTable.SetSessionInfo “user”, “password”
Next
```

Working with secure data in reports

If your report connects to a secure data source that requires log on information, you will not be able to log off of the data source until the Report object has been closed or has gone out of scope. Keep in mind that if you assign the Report object to the ReportSource property of the CRViewer object in the Crystal Reports Report Viewer, the Report object cannot be closed until you assign a new value to the Report Viewer or close the CRViewer object. Thus, you will not be able to log off of a secure data source until the report is no longer displayed in the Report Viewer.

Working with Formatting

Using Report Variables

Certain events, like the OnSectionFormat event, may get fired more than once in a section. For example, a Details section may fire multiple times for a given record if the program is performing "Keep Section Together" calculations for the report. When you are using VB variables that are supposed to increment once for each record (as when you are calculating a running total), you can have problems maintaining state with these multiple firings.

To prevent these problems, the RDC now enables you to declare Report Variables. Report variables are Crystal variables that you can use in your VB code. Report Variables "understand" what's going on in the report and they increment appropriately. This enables you to do more of your coding than ever in VB and still get the results you expect.

There are three methods you use with report variables.

AddReportVariable

You use AddReportVariable to declare a Report Variable. The following code declares four Report Variables:

- CurrencyVal of the type currency
- LongVal of the type number
- DoubleVal of the type number
- StringVal of the type string.

```
AddReportVariable crRVCurrency, "CurrencyVal"
AddReportVariable crRVNumber, "LongVal"
AddReportVariable crRVNumber, "DoubleVal"
AddReportVariable crRVString, "StringVal"
```

GetReportVariableValue

You use GetReportVariableValue to reference the value in a Report Variable. The following code references the value in the Report Variable, CurrencyVal.

```
CurVal = GetReportVariableValue("CurrencyVal")
```

SetReportVariableValue

You use SetReportVariableValue to increment a Report Variable. The following code sets the value of the Report Variable CurrencyVal to the value of the expression “CDbl(CurVal) + CDbl(FieldCurrency.Value)”.

```
SetReportVariableValue "CurrencyVal", CDbl(CurVal) +
CDbl(FieldCurrency.Value)
```

Note that you can only use Report Variables at runtime and that you must use double quotes every time you reference such a variable.

Formatting a section or report object conditionally

You can now do conditional formatting using VB code. Crystal Reports has allowed this in the past with its own formula language but now you can do it in VB code as well.

This Format event is fired each time this section is formatted, so you can get the values of the object and perform a variety of formatting functions.

This first example changes the backcolor of the section itself if the value of the selected field is less than 50000.

```
Private Sub Section6_Format(ByVal pFormattingInfo As Object)
If Field9.Value < 50000 Then
        Section6.BackColor = vbRed
    Else
        Section6.BackColor = vbWhite
    End If
End Sub
```

The second example changes the color of the text in the selected field if the value of that field is less than 50000.

```
Private Sub Section6_Format(ByVal pFormattingInfo As Object)
If Field9.Value < 50000 Then
        Field9.TextColor = vbRed
    Else
        Field9.TextColor = vbGreen
    End If
End Sub
```

Working with Formulas/Selection Formulas

Passing a Selection Formula at Runtime

When you pass the selection formula to Crystal Reports, you use the RecordSelectionFormula property of the Report object.

Note: Formulas must conform to Crystal Reports' formula syntax.(i.e. strings embedded in single quotes while numbers not, etc.). For Crystal Reports to parse a formula it receives from the application (Visual Basic in this case), the formula must be identical to a formula that you would enter directly into the report designer. You can use the MsgBox function in Visual Basic to verify that the formula string being sent from VB is in a format that can be used by Crystal Reports.

Two ways you can use the RecordSelectionFormula property of the Report object are:

- Hardcoding: you can copy your code from your report and paste it into the code.
- Using a variable: you can append the value of a variable onto your code when calling the RecordSelectionFormula property.

```
CRXReport.RecordSelectionFormula = "{Customer.Region} = ‘CA’"
```

Note: Any dates passed to Crystal Reports must be in Crystal Reports Date format, Date(yyyy,mm,dd).

Passing a Formula at runtime

When passing a value to Crystal Reports, you use the.Text property of the FormulaFieldDefinition object.

First, you create your formula using the Crystal Reports formula language.

- If you are giving it a string value, you must place a string value in it. This will force the program to treat the formula as a string.
- A single space inside double quotes is a good idea, since this will not print if left unchanged—yet it holds the string type for when you change the actual value later from VB.

Place the new formula on your report where you would like to display the passed value.

If you will be sending a numeric value or date, again place an appropriate value in the formula in order to establish the data type of the formula.

- To pass a numeric value to a formula, place 0 in the formula when you create the formula in Crystal Reports.
- To pass a date value to a formula, place the Today function in the formula.

The following code shows how you can send a title to your report from your VB application. As instructed above, a formula is inserted at the top of the report, and is named Title. It presently contains a space inside quotes, as in " ".

```
Sub Command1_Click ()
'if we hard-code the title
CRXReport.FormulaFields.Item(1).text = "'Report Title'"
```

```
'If we use the value of the Textbox
Text1.Text = "This is a Title"
CRXReport.FormulaFields.Item(1).text = chr(34) & Text1.text & chr(34)
End Sub
```

The following code changes the value of a numeric formula at runtime.

```
Sub Command1_Click ()
'if we hard-code the numeric value
CRXReport.FormulaFields.Item(1).text = "9"
'If we use the value of a variable
x = 65
CRXReport.FormulaFields.Item(1).text = x
End Sub
```

Changing a subreport’s record selection formula at runtime

Changing the record selection formula for a subreport is similar to changing it for the main report. Before you can change the record selection formula for a subreport, you first must open the subreport. Please refer to “Referencing objects in a subreport” on page 68 for examples of different ways you can access a subreport.

Once you have opened the subreport object, you can use code similar to this to change its record selection formula.

```
CRXSubreport.RecordSelectionFormula = "{Customer.Last Year's Sales} > 100000"
```

The subreport acts like any report, where any object, property, and method that is accessible by the Report object is the same for a subreport. The only limitation is that subreports cannot be previewed, printed, or exported as an individual report.

Referencing a report formula using the formula name

The following code sample shows how to reference a report formula at runtime using the formula name instead of referencing the formula by index.

You can put this code in the General Declarations.

```
Dim CRXReport As New CrystalReport1
Dim CRXFormulaFields As CRAXDRT.FormulaFieldDefinitions
Dim CRXFormulaField As CRAXDRT.FormulaFieldDefinition
```

You can put this code in the Form_Load event procedure.

```
Private Sub Form_Load()
Set CRXFormulaFields = CRXReport.FormulaFields
```

Now you can cycle through the formula fields collection.

```
For Each CRXFormulaField In CRXFormulaFields
```

Here you find the formula you require by specifying its name. Note the format that the .Name property returns.

```
If CRXFormulaField.Name = "{@formula}" Then
```

Now you set the formula text.

```
CRXFormulaField.Text = "{Customer.Name}"
End If
Next
```

Finally, you can view the report.

```
Crviewer1.ReportSource = CRXReport
Crviewer1.ViewReport
End Sub
```

Working with Grouping

Changing the existing group's condition field

For this scenario, assume that a report has been created which is currently grouped on the City field (For example, }Customer.City}). To change the condition field for the existing field at runtime, you can use the following code.

Note: This code assumes that you know which field you want to change and that there is only one group in the report.

You can place this code in the General Declarations section.

```
Dim CRXReport As New CrystalReport1 'The existing report (ActiveX Designer)
Dim CRXDBField As CRAXDRT.DatabaseFieldDefinition
```

You can put this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

Currently the group is based on the CITY field, but you want to change it to the REGION field. This code accesses the first table to get the 12th field which is the REGION field.

```
set CRXDBField = CRXReport.Database.Tables.Item(1).Fields.Item(12)
```

Now you can set the new condition field for the existing group. Since you know that there is only one group in the report, you can reference it by its name "GH".

```
CRXReport.Areas.Item("GH").GroupConditionField = CRXDBField
```

Finally, display the report.

```
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
```

End Sub

Adding a new group to the report

For this scenario, assume that a report has been created and it does not contain any groups. To add a new group to the report at runtime, you can use code similar to the following:

Note: Please note that using this method will not display the group name on the report.

You can place this code in the General Declarations section.

```
Dim CRXReport As New CrystalReport1 'The existing report (ActiveX Designer)
Dim CRXDBField As CRAXDRT.DatabaseFieldDefinition
```

You can place this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

For this example, you want to add a group that is based on the REGION field. This code accesses the first table to get the 12th field, which is the REGION field.

```
set CRXDBField = CRXReport.Database.Tables.Item(1).Fields.Item(12)
```

Now you can add the new group to the report. Please note that:

- 0 is the GroupNumber in case you add more than one group
- `CRXDBField` is the ConditionField
- crGCAnyValue is the condition on which the group changes
- crAscendingOrder is the SortDirection

```
CRXReport.AddGroup 0, CRXDBField, crGCAnyValue, crAscendingOrder
```

Finally, display the report.

```
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
End Sub
```

Adding a group sort field

This scenario requires a report that already contains a group which also must contain a summary field.

A Group Sort Field can only exist if the group contains a summary field, because that summary field is what the Sort is based on. In this example, the report is grouped on {Customer.Region} and the summary field is the "SUM of Customer.Last Year's Sales (Currency)".

To add a group sort field you can use code similar to the following:

1 Place this code in the General Declarations section.

```
Dim CRXReport As New CrystalReport1 'The existing report (ActiveX Designer)
Dim CRXSortFields As CRAXDRT.SortFields
Dim CRXSummaryField As CRAXDRT.SummaryFieldDefinition
```

2 Place this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

3 This code gets the SortFields collection for the groups:

```
Set CRXCRXSortFields = CRXReport.GroupSortFields
```

4 Get the first Summary Field which is the '"SUM of Customer.Last Year's Sales (Currency)".

```
Set CRXSummaryField = CRXReport.SummaryFields.Item(1)
```

5 Now add the Group Sort Field.

```
CRXSortFields.Add CRXSummaryField, crDescendingOrder
```

6 Finally, display the report.

```
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
```

End Sub

Modifying a group's sort direction at runtime

You can change a group's sort direction through the SortDirection property of the Area Object. To change a group's sort direction you can use code similar to the following:

Note: All the samples are based on the sample database, XTREME.MDB, which is installed by Crystal Reports.

1 Put this code in the General Declarations section.

```
Dim CRXReport As New CrystalReport1
```

2 Put this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

The Item parameter "GH1" refers to the first group header; to access the second group header you would use "GH2" This line of code sets the SortDirection to descending.

```
CRXReport.Areas.Item("GH1").SortDirection = crDescendingOrder
```

3 Finally, view the report.

```
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
End Sub
```

Working with parameter fields

Crystal parameters and stored procedure parameters and are both set through the ParameterFieldDefinition Object. The following sample code passes four parameters to a report. The Main Report and the Subreport each have a Stored Procedure and a Crystal Parameter.

This code can go in the General Declarations Section.

```
Private Sub Command1_Click()
Dim CRXParamDefs As CRAXDRT.ParameterFieldDefinitions
Dim CRXParamDef As CRAXDRT.ParameterFieldDefinition
Dim CRXSubreport As CRAXDRT.Report
Set CRXParamDefs = CRXReport.ParameterFields
```

This code cycles through the ParameterFieldDefinitions collection in the main report.

```
For Each CRXParamDef In CRXParamDefs
With CRXParamDef
Select Case .ParameterFieldName
```

It finds and sets the appropriate Crystal parameter.

```
Case "MainParam"
.SetCurrentValue "Main Report Parameter"
```

Now it finds and sets the appropriate stored procedure parameter.

```
Case "[CustomerID]"
.SetCurrentValue "Alfki"
End Select
End With
Next
```

It opens the appropriate subreport.

```
Set CRXSubreport = CRXReport.OpenSubreport("sub1")
Set CRXParamDefs = CRXSubreport.ParameterFields
```

It cycles through the ParameterFieldDefinitions collection for the subreport.

```
For Each CRXParamDef In CRXParamDefs
With CRXParamDef
MsgBox .ParameterFieldName
Select Case .ParameterFieldName
```

It finds and sets the appropriate Crystal parameter for the subreport.

```
Case "SubParam"
.SetCurrentValue "Subreport Parameter"
```

Now it finds and sets the appropriate stored procedure parameter for the subreport.

```
Case "[CustomerID]"
.SetCurrentValue "Anton"
End Select
End With
Next
```

Finally, it disables parameter prompting so the user won't be prompted for a value.

```
CRXReport.EnableParameterPrompting = False
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
End Sub
```

Working with OLE objects

Setting the location of an OLE object

This example demonstrates how you can set the location of an OLE object at Runtime using the SetOleLocation command. The example takes three different OLE objects:

- a bitmap
- part of an Excel worksheet
- part of a Word document.

It cycles through them so the RDC prints a different object each time it formats the Details section.

```
Option Explicit
Private Sub Report_Initialize()
    Database.Tables(1).Location = App.Path + "\Author.mdb"
End Sub
Private Sub SectionDetails_Format(ByVal pFormattingInfo As Object)
```

Object to hold the bitmap:

```
Dim bmpHold As StdPicture
    Dim iModNum As Integer
```

Calculate an integer to pass in as the object name.

```
    iModNum = cRecNum.Value Mod 3 + 1
```

Take an action based on the integer passed in.

```
    Select Case iModNum
```

The bitmap object:

```
    Case 1
```

Set the variable to a bitmap.

```
        Set bmpHold = LoadPicture(App.Path & res\SampleBitmap1.bmp")
```

Set the height and width of the Report object equal to the actual values for the bitmap - the StdPicture object defaults to HiMetric, the Report uses twips.

Set the bitmap on the Report equal to the variable.

```
        Set cOLEObj.FormattedPicture = bmpHold
```

Convert from HiMetric to twips.

```
        cOLEObj.Height = bmpHold.Height * 567 / 1000
        cOLEObj.Width = bmpHold.Width * 567 / 1000
```

The Excel Worksheet object:

```
               Case 2
        cOLEObj.SetOleLocation App.Path & "\res\SampleExcel1.xls"
        cOLEObj.Height = 1800
        cOLEObj.Width = 5791
```

The Word Document object:

```
Case 3
        cOLEObj.SetOleLocation App.Path & "\res\SampleWord1.doc"
        cOLEObj.Height = 322
        cOLEObj.Width = 8641
    End Select
End Sub
```

Working with Sorting

Changing the existing sort's condition field

For this scenario, a report is created which is currently sorted on the Customer Name field (For example, {Customer.Customer Name}). To change the condition field for the existing field at runtime, the following code can be used:

```
'General Declarations
Dim CRXReport As New CrystalReport1
Dim CRXDatabaseField As CRAXDRT.DatabaseFieldDefinition
Private Sub Form_Load()
```

Currently the sort is based on the Customer Name field and the application is to change it to the Last Year's Sale's field. This field must be present on the report. The code accesses the first table to get the 8th field.

```
Set CRXDatabaseField = CRXReport.Database.Tables.Item(1).Fields.Item(8)
```

There is only one Sort on this report so we will access the first item of the SortFields collection and set the field to the Last Year's Sale's field.

```
CRXReport.RecordSortFields.Item(1).Field = CRXDatabaseField
CRViewer1.ReportSource = CRXReport
CRViewer1.ViewReport
End Sub
```

Adding a new sort field to a report

In this example, we’re going to add a sort field that currently isn’t sorted.

You can put code like this in the General Declarations section.

```
Dim CRXReport As New CrystalReport1
Dim CRXDatabaseField As CRAXDRT.DatabaseFieldDefinition
Private Sub Form_Load()
```

Currently there is no sort in this Report. To add the sort field Customer Name, the application must first get the {Customer.Customer Name} field from the Customer Table. This code accesses the first table to get the 2nd field.

```
Set CRXDatabaseField = CRXReport.Database.Tables.Item(1).Fields.Item(2)
```

Now add the field to the SortFields Collection and set the Sort Order to ascending.

```
CRXReport.RecordSortFields.Add CRXDatabaseField, crAscendingOrder
```

Note: If the SortField is added while the Report is viewing, you will have to refresh the viewer before the new sort will be active. This can be done by clicking the Refresh button in the Report Viewer or refreshing the viewer through code. For example,

```
    CRViewer1.Refresh
    CRViewer1.ReportSource = CRXReport
    CRViewer1.ViewReport
    End Sub
```

Working with Summary Fields

How to change summary field kind

This example populates a combo box with the various summary field kind options and then sets a summary field to the kind selected. This code assumes that you have elsewhere declared a variable cr for your report.

```
Private Sub Form_Load()
Dim i As Integer
```

Populate the combo box.

```
cmbSummaryType.AddItem "crSTSum"
cmbSummaryType.AddItem "crSTAverage"
cmbSummaryType.AddItem "crSTSampleVariance"
cmbSummaryType.AddItem "crSTSampleStandardDeviation"
cmbSummaryType.AddItem "crSTMaximum"
cmbSummaryType.AddItem "crSTMinimum"
cmbSummaryType.AddItem "crSTCount"
cmbSummaryType.AddItem "crSTPopVariance"
cmbSummaryType.AddItem "crSTPopStandardDeviation"
cmbSummaryType.AddItem "crSTDistinctCount"
```

Get the SummaryFields collection.

```
Set CRXSummaryFieldDefinitions = CRXReport.SummaryFields
txtCount = CrystalSummaryFieldDefinitions.Count
```

Cycle through the collection.

```
For i = 1 To CRXSummaryFieldDefinitions.Count
    Set CRXSummaryFieldDefinition =
CRXSummaryFieldDefinitions.Item(i)
    lstFieldDefinition.AddItem CRXSummaryFieldDefinition.Name
Next i
End Sub
```

Change the summary field kind.

```
Private Sub lstFieldDefinition_Click()
     Set CRXSummaryFieldDefinition =
CRXSummaryFieldDefinitions.Item(lstFieldDefinition.ListIndex + 1)
     With CRXSummaryFieldDefinition
        cmbSummaryType.Text = cmbSummaryType.List(.SummaryType)
        txtKind = .Kind
        txtName = .Name
        txtNumberofBytes = .NumberOfBytes
        txtValue = OnFormat
        txtValueType = .ValueType
      End With
End Sub
```

Working with Text Objects

Simple and complex text objects

There are two types of text objects: simple and complex.

- Simple text objects are processed internally by the report engine.
- Complex complex text objects are handled by CRPAIG32.DLL.

Only complex text objects can handle carriage returns and line feeds.

If you want to use carriage returns and line feeds in your text object, you will need to force the text object to be complex. To do this, double-click on the text object to edit it and enter a carriage return (press the enter key) inside of it. Then, when carriage returns and line feeds are passed to the text object through your code, the RDC will process them properly.

Changing the contents of a Text Object

You can use text objects in a variety of different ways:

- You can use a text object in a simple way to personalize a report based on your users' input (changing the report title, for example).
- You can also use a text object to display the results of complex operations on database data.

Text objects are very flexible and powerful reporting tools.

To change the contents of a text object you use the SetText method.

- The Text property of the object model's TextObject gives you information about the existing value; it is a read-only property however.
- To assign a new value to the property, you must use the SetText method.

Note: The following examples change the contents of a Text object during the Load event of the Form containing the Crystal Report Viewer ActiveX control. This is especially important if the value of your text object is dependent on data in your data source. Since this is also a common place to change the data source used by a report, changing values in Text objects during the Load event often becomes convenient.

Making a simple text change

This example simply assigns a string to the Text1 object in the report.

```
Private Sub Form_Load()
    Dim CRXReport As New CrystalReport1
    ‘ More code here
    CRXReport.Text1.SetText “Here is some text from my app”
    CRViewer1.ReportSource = CRXReport
    CRViewer1.ViewReport
End Sub
```

Displaying a calculated result

A more complicated technique may be to access data in a database, calculate a new value in Visual Basic, and then display that value through a Text object on the report. In such cases, you can supply a text object for each result when you design your report in RDC Designer. The following code assumes that you have done this:

```
Private Sub Form_Load()
```

Set up the variables.

```
    Dim maxValue As Currency
    Dim maxValueString As String
    Dim CRXReport As New CrystalReport1
    Dim rs As New ADODB.Recordset
```

Set your data source.

```
rs.Open "SELECT * FROM Customer", _
"DSN=Xtreme Sample Database;", adOpenKeyset
CRXReport.Database.SetDataSource rs
maxValue = 0
```

Read the records and calculate the results.

```
    rs.MoveFirst
    Do While Not rs.EOF
        If rs.Fields("Last Year's Sales") > maxValue Then
            maxValue = rs.Fields("Last Year's Sales")
        End If
        rs.MoveNext
    Loop
```

Format and assign the results to the text object.

```
    maxValueString = "The maximum value is " & _
        Format(maxValue, "Currency")
    CRXReport.Text1.SetText maxValueString
```

Preview the report.

```
    CRViewer1.ReportSource = CRXReport
    CRViewer1.ViewReport
End Sub
```

In this example, we are finding the maximum value of the Last Year's Sales field from the new data source, formatting that value as Currency, then displaying a message containing the value through a Text object on the report. As you can see, Text objects provide many options for controlling the output of data in your reports.

Programming the Crystal Report Viewers 8

The Crystal Report Viewer is a front-end user interface for viewing reports. In this chapter you will find detailed information on implementing the ActiveX and Java Bean viewers in your application.

Enhancements to the Report Viewer

The Report Viewer has been enhanced substantially in Version 8:

- The Report Viewer uses multi-threading. As a result, your users can begin viewing a report sooner, even if the report requires that it all be run before certain values are generated (page numbering, for example, of the style "page 24 of 125"). In such a case, the Report Engine uses place holders for the yet-to-be-generated total page count. When that page count is completed, the Report Engine inserts the missing data into the pages already read.
- The report's group tree is loaded on-demand. This allows your users to use the tree functionality for navigation even when only a partial group tree has been loaded.
- You can specify a page number to go to in the report you are currently viewing.
- You can use the Select Expert and the Search Expert in the viewer to select records and search for specific values using formulas.
- The Report Viewer supports the use of rotated text in the report.
- The toolbar for the Report Viewer for ActiveX has a new look.
- You can customize the Report Viewer by resizing sections of the toolbar, adding custom bitmaps, and more.
- There is a Help button implemented for applications. Clicking on the Help button can fire an event to your application so it can display the appropriate help.
- There are over 30 events giving you the ability to make previewing the report a truly interactive activity.

For a better understanding of all the capabilities of the Report Viewer, review the viewer object model (CRVIEWERLibCtl) in the Visual Basic Object Browser.

Note: Visit the Seagate Software Developer Zone web site at
`http://www.seagatesoftware.com/products/dev_zone`.
Click Support to get links for finding documentation and knowledge base articles about the Report Designer Component.

Application Development with Crystal Report Viewers

Developing applications that display reports on screen is now a straightforward process. Crystal Reports includes the Crystal Report Viewers as easy to use but complex components that can be embedded directly in an application. Once added to an application, reports accessed through the Report Engine Automation Server, the Report Designer Component, or the Crystal Web Reports Server can be displayed right inside your own applications. The Report Viewer retains all of the powerful formatting, grouping, and totalling power of the original report, and your users get access to data in a dynamic and clear format.

Crystal Reports provides two Report Viewers specifically designed for application development: the Crystal Report Viewer for ActiveX and the Crystal Report Viewer Java Bean. Both provide a complete object model for programming and manipulating the Report Viewer at runtime inside your applications. Simply displaying a single report inside the Report Viewer is a simple process requiring only a couple of lines of code. However, if necessary for your application, you have the option of complete control over how the Report Viewer appears and functions.

With the Crystal Report Viewer as a front-end user interface for viewing reports, Crystal Reports development technologies allow you to develop even complex client/server or multi-tier applications that access, manipulate, and display data for intranet systems, workgroups, or any group of people needing clear and informative data on a regular basis. Design robust Business Support systems and Enterprise Information Management applications delivering even the most complex data through the Crystal Report Viewers.

This chapter describes both the ActiveX and Java Bean versions of the Report Viewer in relation to designing applications using Crystal Reports development technologies.

Crystal Report Viewer for ActiveX

The Crystal Report Viewer for ActiveX is a standard ActiveX control that can be added to an application in any development environment that supports ActiveX. Programmers using Visual Basic, Delphi, Visual C++, or Borland C++ programmers all receive the benefit of quickly adding a powerful report viewer to an application with little coding.

As a standard component, the ActiveX Report Viewer exposes several properties at design time, but also provides a complete object model with properties, methods, and events that can be programmed at runtime. The following sections discuss various topics for working with the ActiveX Report Viewer in Visual Basic. If you are using a development environment other than Visual Basic, use these topics as a guideline, but refer to your development software documentation for specific information on working with ActiveX controls.

The Crystal Report Viewer, as an ActiveX control, includes a complete object model for controlling how it appears in an application, and how it displays reports. Simply displaying a report in the Report Viewer window takes little code, but to truly make use of its power requires a broader understanding of how to work with the object model.

Related topics:

“Adding the Report Viewer to a Visual Basic project” on page 92.

“Using the CRViewer object” on page 92.

Adding the Report Viewer to a Visual Basic project

If you create a new report using the Create Report Expert in the Crystal Report Designer Component, the Report Viewer control can be automatically added to a Form in your Visual Basic project. However, there may be times when you need to add the control by hand. In addition, the Report Viewer control can be implemented in other environments, many of which may not support ActiveX designers, meaning the Create Report Expert is unavailable.

Use the following steps to add the Crystal Report Viewer ActiveX control to a Form in your Visual Basic application. This tutorial assumes the Form already exists in your project and is named Form1.

1. First, you must verify that a reference to the Report Viewer control exists in your project. From the Project menu, select the Components command. The Components dialog box appears.
2. On the Controls Tab of the Components dialog box, scroll through the list of ActiveX controls until you find *Crystal Report Report Viewer*.

Note: If you do not see the Crystal Report Report Viewer control in the list, use the Browse button to locate the CRVIEWER.DLL component in the C:\Program Files\Seagate Software\Viewers\ActiveXViewer directory.

3. If the check box next to the Report Viewer control is not toggled on, toggle it on now.
4. Click OK, and the CRViewer control will appear in the Visual Basic toolbox.
5. Click the CRViewer control on the toolbox, then draw the Report Viewer control on your form by dragging a rectangle across the form with the mouse pointer. An instance of the control will be added to your Form.
6. Adjust the size and position of the Report Viewer on your form, and use the Properties window to adjust the overall appearance of the control.

Using the CRViewer object

The CRViewer object represents an instance of the Report Viewer control that has been added to your project. If you have created a report using the Crystal Report Designer Component and accepted the defaults for adding the Report Viewer to your project, the Report Viewer control in your application will be named CRViewer1. CRViewer1 can be used in your code as a CRViewer object. For instance, the following code demonstrates a simple technique for assigning a report to the Report Viewer, and displaying it:

```
CRViewer1.ReportSource = report
CRViewer1.ViewReport
```

For more information on the properties and methods available with this object, refer to the Report Viewer object model and the CRViewer object.

The topics listed below describe several aspects of the Report Viewer object model and present examples of how to use the Report Viewer objects, methods, properties and events in your Visual Basic code.

Specifying a report

The most important task with the Report Viewer control is to specify a report and display it at runtime. This is easily handled with the ReportSource property and the ViewReport method.

```
Private Sub Form1_Load()
 Dim report As New CrystalReport1
 CRViewer1.ReportSource = report
 CRViewer1.ViewReport
End Sub
```

In this example, assigning the report and displaying it in the Report Viewer is handled when the Form containing the Report Viewer object is loaded into the application. A reference to the report is first obtained in the form of a Report object representing a Crystal Report Designer Component that has been added to the Visual Basic project.

ReportSource is a property of the Report Viewer's CRViewer object which corresponds directly to the Report Viewer control added to the project. In this case, that control has been named *CRViewer1*. The ReportSource property can accept a report in the form of a Report Object exposed by the Report Designer Component or the Crystal Web Reports Server.

Finally, the ViewReport method is called. This method has no parameters and has the job simply of displaying the specified report inside the Report Viewer control.

Working with secure data in reports

If your report connects to a secure data source that requires log on information, you must release the Report object from the Report Viewer before you can log off of the data source. This can be done by assigning a new Report object to the ReportSource property, or by closing the CRViewer object. Until this is done, the data source will not be released from the Report object and you cannot log off.

Handling Report Viewer events

The Report Viewer control allows you to write custom code for several events relating to user interaction with both the control window and the report displayed. For instance, if you design a drill down report using the Report Designer Component, your users are likely to want to drill down on detail data. You can provide custom handling of such an event by writing code fort the DrillOnGroup event.

To add event procedures to the Report Viewer control for the DrillOnGroup and PrintButtonClicked events:

1. In the Visual Basic Project window, select the Form containing the Report Viewer control.
2. Click the View Code button in the toolbar for the Project window. A code window for the form appears.
3. In the drop-down list box at the upper left hand corner of the code window, select the CRViewer1 control. (This name will appear different if you changed the Name property of the control in the Properties window.)
4. In the drop-down list box at the upper right corner of the code window, select the DrillOnGroup event. A procedure appears for handling the event.
5. Add the following code to the DrillOnGroup event procedure:

```
Private Sub CRViewer1_DrillOnGroup(GroupNameList As Variant, _
    ByVal DrillType As CRVIEWERLibCtl.CRDrillType, UseDefault As Boolean)
    MsgBox "You're drilling down on the " & GroupNameList(0) & " group!"
End Sub
```

6. In the drop-down list box at the upper right of the code window, select the PrintButtonClicked event. A new procedure appears for this event.
7. Add the following code for the new event:

```
Private Sub CRViewer1_PrintButtonClicked(UseDefault As Boolean)
    MsgBox "You clicked the Print button!"
End Sub
```

The DrillOnGroup event is triggered when a user double-clicks on a chart, on a map, or on a report summary field. The code added to the event procedure will display a message box with the name of the group. The PrintButtonClicked event is fired if the user clicks the print button on the Report Viewer window. Note that any code added to these event handlers replaces the default action of the event. A more practical use of these events would be to display custom dialogs or perform other report related calculations and procedures.

Moving through a report

Often, reports consist of several pages. The Report Viewer control provides, by default, controls that allow a user to move through the pages of the report. However, you may need to implement a system through which your own code controls when separate pages are displayed.

The CRViewer object provides several methods for moving through a report, including methods to move to specific pages:

- ShowFirstPage
- ShowLastPage
- ShowNextPage
- ShowPreviousPage
- ShowNthPage
- GetCurrentPageNumber

And methods for moving to specific groups in the report:

- ShowGroup

Moving through pages

The first set of methods designed for moving through the pages of a report are straightforward and correspond directly to controls that normally appear on the Report Viewer control window. ShowFirstPage, ShowLastPage, ShowNextPage, and ShowPreviousPage simply switch to the first, last, next, or previous page in the report, respectively. They are all used in the same manner in code:

```
CRViewer1.ShowFirstPage
CRViewer1.ShowLastPage
CRViewer1.ShowNextPage
CRViewer1.ShowPreviousPage
```

If the requested page cannot be displayed, for instance, if the last page in the report is currently displayed and ShowNextPage is called, the currently displayed page will be refreshed.

For more controlled movements through the report, ShowNthPage can display a specific page of the report:

```
CRViewer1.ShowNthPage 5
```

This method accepts a page number as its only argument. If the selected page number does not exist, for example, page 10 is selected from a 6 page report, then either the last or first page will be displayed, depending on the page number requested.

As a convenience, the GetCurrentPageNumber method has also been included. You can obtain the currently displayed page from within your code at any time using this method:

```
Dim pageNum As Long
pageNum = CRViewer1.GetCurrentPageNumber
```

Moving to a specific group

Grouping is a common feature of reports, and, since page numbers can frequently change based on current data, it may be more appropriate to navigate through a report using groups. For example, if a report is grouped by cities within states, and by states within countries, you can include code to display the group for a specific city.

Printing the report

Although the Report Viewer control is designed primarily for displaying reports on screen, users frequently want a hard-copy of the data. The PrintReport method provides a simple means of allowing access to the Windows print features. Simply call the method as below, and Windows can take over.

```
Dim Report As New Crystalreport1
CRViewer1.ReportSource = Report
CRViewer1.PrintReport
```

Controlling the appearance of the Report Viewer

By default, the Report Viewer window includes several controls for allowing users to navigate through a report, enlarge the view of a report, refresh the data in a report, and more. There may be applications that you create in which you want to limit a user's interaction, change the look of the Report Viewer window, or provide an alternate means of accessing the same functionality.

For instance, you could turn off the navigation controls in the Report Viewer, then create your own controls to navigate through the report that call the ShowFirstPage, ShowLastPage, ShowNextPage, ShowPreviousPage, and ShowNthPage methods. (See "Moving through a report" on page 95.) For handling such custom features, the Report Viewer object model provides several properties for enabling and disabling different features of the Report Viewer ActiveX control:

- DisplayBackgroundEdge
- DisplayBorder
- DisplayGroupTree
- DisplayTabs
- DisplayToolbar
- EnableAnimationCtrl
- EnableCloseButton
- EnableDrillDown
- EnableGroupTree
- EnableNavigationControls
- EnablePrintButton

- EnableProgressControl
- EnableRefreshButton
- EnableSearchControl
- EnableStopButton
- EnableToolbar
- EnableZoomControl

Using these properties requires assigning a value of either True or False. True enables the specified control or feature of the Report Viewer, while False disables it. All controls and features are, by default, enabled.

The following code demonstrates how to disable the entire toolbar for the Report Viewer window:

```
CRViewer1.DisplayToolbar = False
```

Connecting to the Web Reports Server

The Web Reports Server provides not only a powerful means of distributing reports across the web, but also provides a report distribution mechanism that can be incorporated into multi-tier applications. By using the Crystal Report Viewer for ActiveX as a client-side report viewer, the Web Reports Server can become a report distribution engine within a larger application that runs over a network.

Connecting to the Web Reports Server requires accessing two new ActiveX components: the WebReportBroker and the WebReportSource. The following samples demonstrate how to connect to the Web Reports Server using "Connecting from Visual Basic" on page 97, and "Connecting from VBScript" on page 98, inside a web page.

Connecting from Visual Basic

The following code is an example of how to connect to the Web Reports Server from Visual Basic and assign a report to the Crystal Report Viewer for ActiveX. This assumes that you have added the ActiveX viewer control to a form named Form1, and the ActiveX viewer control is named CRViewer1.

```
Private Sub Form1_Load()
    Dim webBroker, webSource
    Set webBroker = CreateObject("WebReportBroker.WebReportBroker")
    Set webSource = CreateObject("WebreportSource.WebReportSource")
    webSource.ReportSource = webBroker
    webSource.URL = "http://<machinename>/scrreports/xtreme/hr.rpt"
    webSource.Title = "Employee Profiles"
    CRViewer1.ReportSource = webSource
    CRViewer1.ViewReport
End Sub
```

Connecting from VBScript

The following code assumes you have added the Crystal Report Viewer for ActiveX to a web page using the <OBJECT> tag and assigned it an ID of CRViewer.

```
<OBJECT ID="WebSource" Width=0 Height=0>
    CLASSID="CLSID:F2CA2115-C8D2-11D1-BEBD-00A0C95A6A5C"
    CODEBASE="viewer/ActiveXViewer/swebrs.dll#Version=1.2.0.5"
</OBJECT>
<OBJECT ID="WebBroker" Width=0 Height=0>
    CLASSID="CLSID:F2CA2119-C8D2-11D1-BEBD-00A0C95A6A5C"
    CODEBASE="viewer/ActiveXViewer/swebrs.dll#Version=1.2.0.5"
</OBJECT>
<OBJECT ID="Export" Width=0 Height=0>
    CLASSID="CLSID:BD10A9C1-07CC-11D2-BEFF-00A0C95A6A5C"
    CODEBASE="viewer/ActiveXViewer/sviewhlp.dll#Version=1.0.0.4"
</OBJECT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Page_Initialize
    Dim webBroker
    Dim webSource
    Set webBroker = CreateObject("WebReportBroker.WebReportBroker")
    Set webSource = CreateObject("WebReportSource.WebReportSource")
    webSource.ReportSource = webBroker
    webSource.URL = Location.Protocol + "//" + Location.Host + _
        "/scrreports/xtreme/invent.rpt"
    CRViewer.ReportSource = webSource
    CRViewer.ViewReport
End Sub
-->
</SCRIPT>
```

The Crystal Report Viewer Java Bean

The Crystal Report Viewer Java Bean (or Report Viewer Bean) can be added to an application in any development environment that supports Java (version 1.1). Programmers receive the benefit of quickly adding a powerful report viewer to an application with little coding.

As a standard component, the Crystal Report Viewer Java Bean exposes several properties at design time, but also provides a complete object model with properties, methods, and events that can be programmed at runtime. The following discusses one approach to creating an application using the Crystal Report Viewer Java Bean. It describes the creation of a simple Applet which will allow a report to be viewed from your browser.

This example uses the Bean Box a component of the Bean Developer Kit (BDK) from Sun Microsystems Inc. The Bean Box is not intended to be used for serious application development, rather as a platform for testing Beans interactively at design time, and creating simple applets for run time testing. The Bean Box is available for download from Sun Microsystems.

Adding the Report Viewer Bean to the project

To add the Report Viewer Bean to the Bean Box:

1. Locate the JAR file called ReportViewerBean.jar in the "Viewers" directory (\SeagateSoftware\Viewers\JavaViewerBean).
2. Either copy the file to the \jars subdirectory of the BDK
   or
   From the Bean Box Select LoadJar from the File menu and specify the pathname of the file.
3. The Crystal Report Viewer Icon should appear in the ToolBox palette.

Creating a simple applet with the Report Viewer

To add the Report Viewer Bean to the Bean Box Composition window and create an applet:

1. Click on the Report Viewer Beans name (Crystal Report Viewer) in the ToolBox palette.
2. Click on the location in the Bean Box Composition window where you want the Report Viewer Bean to appear.
3. Resize the Report Viewer in the Composition window until you are able to see the controls and report window.
4. In the Bean Box Property Sheet window you will see the list of Report Viewer Bean properties. These can be set and edited. For example to view a report click on the reportName property. When the dialog box appears enter the URL of a report file (for example: "http://localhost/scrreports/craze/adcont2s.rpt").
   The report should be displayed in the Crystal Report Viewer Report window.
5. To create a simple applet select MakeApplet from the File menu. This will create an applet which when called from your browser will display the report specified in the reportName property. You will be prompted to specify a directory where your applet and its supporting file will be placed (or the default tmp subdirectory of the beanbox directory).

If you look at the directory containing the applet, you will notice that there are a number of supporting files and directories. Locate the html file (<appletname>.html) and click on it. Your default browser should display the Report Viewer and the report.

The minimum required to actually run the application using the bean is:

- the html file which references the applet class file
- the extracted ReportViewerBean.jar file and any supporting jar files
- the applet class file .

Programming the Embeddable Crystal Reports Designer Control

9

This chapter demonstrates how to integrate the Embeddable Crystal Reports Designer Control (Embeddable Designer) into your application. Included are tutorials on creating sample applications in both Microsoft Visual Basic and Microsoft Visual C++. A brief section on creating reports in the designer at runtime is also included.

Overview

The Embeddable Crystal Reports Designer Control (Embeddable Designer) is a new addition to the Report Designer Component. Easily integrated into your application, the Embeddable Designer provides your users with an interactive environment to design or edit Crystal Reports.

The following examples demonstrate how to create a sample application using the Embeddable Designer in both Microsoft Visual Basic and Microsoft Visual C++. Two tutorials are provided:

- "Creating a Microsoft Visual Basic sample application with the Embeddable Designer" on page 102
- "Creating a Microsoft Visual C++ sample application with the Embeddable Designer" on page 108

Creating a Microsoft Visual Basic sample application with the Embeddable Designer

This tutorial assumes that you have prior knowledge of Microsoft Visual Basic, Crystal Reports, and the Report Designer Component. The following procedure and sample code detail how to create an application that demonstrates the Embeddable Designer. The application will consist of a single form with a tab control and buttons to create a new report or open an existing report. The tab control has two property pages. The first page (Designer tab) contains the Embeddable Designer, and the second page (Viewer tab) contains the Crystal Reports Viewer Control. See "Programming the Crystal Report Viewers" on page 89. With the application running, you will be able to open a new report or an existing report in the Embeddable Designer. After designing or editing a report, view it in the Crystal Report Viewer. When you go back to the Embeddable Designer to make changes, you'll see those changes updated automatically in the Crystal Report Viewer.

Note: The sample application was created in Microsoft Visual Basic version 6.0 with Service Pack 4.

This tutorial contains the following steps:

- "Step 1: Creating the user interface" on page 103
- "Step 2: Writing the code" on page 104
- "Step 3: Running the Embeddable Designer application" on page 107.

Step 1: Creating the user interface

In this step, you will add the references, components, forms, and controls for the application.

1. Create a new **Standard EXE** project.
2. Add the following components to the project:
   - Crystal Reports Viewer Control
   - Embeddable Crystal Reports 8.5 Designer Control
   - Microsoft Common Dialog 6.0
   - Microsoft Tabbed Dialog Control 6.0 (Sp4).
3. Add the following reference to the project:
   - Crystal Reports 8.5 ActiveX Designer Design and Runtime Library.

   Note: See "Programming the Crystal Report Viewers" on page 89 for more information.
4. Add the following controls from the Toolbox to the form:
   - three CommandButton controls
   - Common Dialog (Microsoft Common Dialog Control 6.0)
   - SSTab1 (Microsoft Tabbed Dialog Control 6.0 (SP4))
   - CRDesignerCtrl (Embeddable Crystal Reports 8.5 Designer Control)
   - CRViewer (Crystal Report Viewer Control).
5. Set the properties for the form and the controls:
   - **Form1** (Main form for the application):
     - Name = frmMain
     - Caption = Embeddable Crystal Reports 8.5 Designer Control Sample
     - Height = 8265
     - Width = 10815
   - **SStab** (Tabbed control containing the Embeddable Designer and Report Viewer controls):

   Note: Click the tabs on SSTab1 to access the separate Caption properties.
     - Name = SSTab1
     - Tabs = 2
     - Tab1 Caption = Design
     - Tab2 Caption = Preview
     - Enabled = False
     - Height = 7600
     - Left = 0
     - Tab Height = 300
     - Top = 130
     - Width = 10825

- **CRDesignerCtrl1** (Designs and edits reports):

Note: Place the control on the tab labeled "Designer."

  - Name = CRDesignerCtrl1
  - Height = 6540
  - Left = 0
  - Top = 0
  - Width = 10325
- **CRViewer** (Views reports):
  - Name = CRViewer1
  - Height = 10340
  - Left = 120
  - Top = 360
  - Height = 6600
- **CommonDialog** (Creates an Open dialog to select reports):
  - Name = CommonDialog1
  - Filter = Crystal Reports |*.rpt
- **Button1** (Creates a new report):
  - Name = cmdNewReport
  - Caption = &New Report
  - Left = 120
  - Top = 7850
- **Button2** (Opens an existing report):
  - Name = cmdOpenReport
  - Caption = &OpenReport
  - Left = 1950
  - Top = 7850
- **Button3** (Closes the form and exits the application):
  - Name = cmdExit
  - Caption = E&xit
  - Left = 9030
  - Top = 7850

Step 2: Writing the code

In this step you will add the code to:

- create a new report
- create an Open dialog box to open an existing report
- set the report object to CRDesignerCtrl1 and view the report in design mode
- set the report object to CRViewer1, set the zoom level, and view the report
- refresh the report when switching from the Designer tab to the Viewer tab on the Microsoft Tabbed Dialog Control.

1 Type or insert the sample code below into the code module of frmMain.

2 Once you have added the code, on the **Run** menu click **Start** to run the application.

Note: Error handling is not included in the code samples.

```
Option Explicit

Dim m_Application As New CRAXDDRT.Application
Dim m_Report As CRAXDDRT.Report

' *************************************************************
'DisplayReport is a procedure that
' - Enables the Tab control the first time a report is created
'   or opened.
' - Sets the report object to the Embeddable Designer(CRDesigner1).
' - Disables the Help menu in the Embeddable Designer.
' - Sets the report object to the Crystal Report Viewer Control
'   (CRViewer1).
' - Sets the Crystal Reports Viewer to view the report.
'
Public Sub DisplayReport()
' Enable the tab control if disabled.
If SSTab1.Enabled = False Then SSTab1.Enabled = True

' Set the Report Object
CRDesignerCtrl1.ReportObject = m_Report

' Note----------------
' Set all other properties for CRDesignerCtrl1 after setting the
' ReportObject property
' --------------------

' Disable the Help menu
CRDesignerCtrl1.EnableHelp = False

' Set the report source
CRViewer1.ReportSource = m_Report

' Set the viewer to view the report
CRViewer1.ViewReport
' Set the zoom level to fit the page
' to the width of the viewer window
CRViewer1.Zoom 1
End Sub

' *************************************************************
Private Sub Form_Load()
'Set the tab control to display the Designer tab
'when the form is loaded
SSTab1.Tab = 0
End Sub
```

```
' *************************************************************
Private Sub SSTab1_Click(PreviousTab As Integer)
' Refresh the report when clicking Preview,
' without refreshing the data from the server.
If PreviousTab = 0 Then CRViewer1.RefreshEx False

End Sub

' *************************************************************
' Create a new report and display it in the Embeddable Designer
'
Private Sub cmdNew_Click()
' Set the report object to nothing
Set m_Report = Nothing

' Create a new report
Set m_Report = m_Application.NewReport

' Call DisplayReport to set the report to the Embeddable Designer
' and the Crystal Report Viewer and then display the report in the
' Embeddable Designer.
Call DisplayReport
End Sub

' *************************************************************
' Use the Microsoft Common Dialog control to open a report.
'
Private Sub cmdOpen_Click()

CommonDialog1.CancelError = True

On Error GoTo errHandler

' Display the open dialog box
CommonDialog1.ShowOpen

' Set the report object to nothing
Set m_Report = Nothing

' Open the selected report
Set m_Report = m_Application.OpenReport(CommonDialog1.FileName, 1)

' Call DisplayReport to set the report to the Embeddable Designer
' and the Crystal Report Viewer
Call DisplayReport

Exit Sub

errHandler:
'User cancelled dialog
End Sub
```

```
' ************************************************************
Private Sub cmdAbout_Click()
frmAbout.Show vbModal
End Sub

' ************************************************************
Private Sub cmdExit_Click()
Unload Me
End Sub
```

Step 3: Running the Embeddable Designer application

In this step you will:

- create and design a new report in the Embeddable Designer
- view the report and any changes made to the report
- open and edit an existing report in the Embeddable Designer
- view the report and any changes made to the report.

1. With the application running, click **New Report**.

   An empty report will appear in the Embeddable Designer.

   Note: The interface for the Embeddable Designer is the same one used for ActiveX Designer reports created in the Microsoft Visual Basic IDE with the Crystal Reports Report Designer Component.

2. Design a new report to view.

   If you are not familiar with the designer environment, see "Designing reports in the Embeddable Designer" on page 117 for a step-by-step procedure on creating a simple report off the Xtreme Sample Database ODBC data source.

3. Click **Preview** to view the report in the Crystal Report Viewer.
4. Click **Design** and make some changes to the report. Then click **Preview** to see the changes in the Crystal Report Viewer.
5. Click **Open**.
6. In the Open dialog box select one of the sample reports and click **Open** to view the report in the Embeddable Designer.
7. Click **Preview** to view the report in the Crystal Report Viewer.
8. Click **Design** and make some changes to the report. Then click **Preview** to see the changes in the Crystal Report Viewer.

Creating a Microsoft Visual C++ sample application with the Embeddable Designer

This tutorial assumes that you have prior knowledge of Microsoft Visual C++, Crystal Reports, and the Report Designer Component. The following procedure and sample code create an application that demonstrates the Embeddable Designer. The application will consist of a main dialogs for the Embeddable Designer, and for the Crystal Reports Viewer Control. For details, see "Programming the Crystal Report Viewers" on page 89.

With the application running, you will be able to open a new or existing report in the Embeddable Designer. After designing or editing a report, you can preview it in the Crystal Report Viewer by clicking the Preview button. You can then return to the Embeddable Designer by clicking the Design button. When you go back to to edit the report, you can see those changes updated automatically by clicking the Preview button again.

This tutorial contains the following steps:

- "Step 1: Creating the User Interface" on page 108
- "Step 2: Adding member variables and member functions" on page 110
- "Step 3: Writing the code" on page 112
- "Step 4: Running the Embeddable Designer application" on page 116.

Step 1: Creating the User Interface

In this step, you will create a Dialog-based MFC.EXE application and add the components, and controls for the application.

1. Using the AppWizard, create a starter Dialog-based MFC application. The main dialog will display reports in Embeddable Designer and the Crystal Report Viewer.
   - Project Name: Embeddable_Designer
   - Application Class Name: CEmbeddable_DesignerApp
   - Dialog Class Name: CEmbeddable_DesignerDlg
2. Add the Embeddable Designer Control and the Crystal Report Viewer Control through the Components and Controls Gallery dialog box. The controls are in the Registered ActiveX Controls folder:
   - Embeddable Crystal Reports 8.5 Designer Control
   - Crystal Report Viewer Control
3. Add the following controls from the Toolbox to **IDD_EMBEDDABLE_DESIGNER_DIALOG** (Embeddable Designer dialog):
   - six buttons
   - Embeddable Crystal Reports 8.5 Designer
   - Crystal Report Viewer

4 Set the properties for the Embeddable Designer dialog and its controls:
- **IDD_EMBEDDABLE_DESIGNER_DIALOG** (Create, open, save, design and preview reports):
  - Caption: Embeddable Designer
  - Height: 340
  - Width: 540
- **Embeddable Crystal Reports 8.5 Designer Control** (Designs and edits reports):
  - Height: 300
  - Left: 7
  - Top: 7
  - Width: 525
- **Crystal Report Viewer:**
  - Height: 300
  - Left: 7
  - Top: 7
  - Width: 525

Note: Either the Embeddable Designer or the Crystal Report Viewer will be hidden depending on wether the user is designing or previewing a report.

- **All Buttons** (Common Properties):
  - Align all Buttons to the bottom of the form
  - Height: 14
  - Width: 50
- **Button1** (Creates a new report):
  - ID: IDC_NEW_REPORT
  - Caption: &New Report
  - Left: 7
- **Button2** (Opens an existing report):
  - ID: IDC_OPEN_REPORT
  - Caption: &Open Report
  - Left: 66
- **Button3** (Saves a report):
  - ID: IDC_SAVE_REPORT
  - Caption: &Save Report
  - Left: 125
- **Button4** (Shows the Crystal Report Viewer and hides the Embeddable Designer.):
  - ID: IDC_PREVIEW
  - Caption: &Preview
  - Left: 184

- **Button5** (Shows the Embeddable Designer and hides the Crystal Report Viewer):
  - ID: IDC_SHOW_DESIGNER
  - Caption: &Design
  - Left: 184
- **Button6** (Exits the application):
  - ID: IDCANCEL
  - Caption: E&xit
  - Left: 483

Step 2: Adding member variables and member functions

In this step, you will add the member variables for the controls on the Embeddable Designer dialog and then add member functions, and member variables for the CEmbeddable_DesignerDlg class.

This step is broken into 2 sections:

- "Adding member variables for the Embeddable Designer dialog controls" on page 110
- "Adding member variables and functions to the CEmbeddable_Designer class" on page 111

Adding member variables for the Embeddable Designer dialog controls

Use the Class Wizard to create the member variables for the controls.

1. Create member variables for the controls on the Embeddable Designer dialog:
   - **Embeddable Designer Control:**
     - Control ID: IDC_EMBEDDABLECRYSTALREPORTSDESIGNERCTRL
     - Type: CCRDesignerCtrl
     - Member: m_Designer
   - **Crystal Report Viewer Control:**
     - Control ID: IDC_CRVIEWER1
     - Type: CCrystalReportViewer4
     - Member: m_Viewer
   - **New Report Button:**
     - Control ID: IDC_NEW_REPORT
     - Type: CButton
     - Member: m_NewReport
   - **Open Report Button:**
     - Control ID: IDC_OPEN_REPORT

    - Type: CButton
    - Member: m_OpenReport
  - **Save Report Button:**
    - Control ID: IDC_SAVE_REPORT
    - Type: CButton
    - Member: m_SaveReport
  - **Preview Button:**
    - Control ID: IDC_PREVIEW
    - Type: CButton
    - Member: m_Preview
  - **Design Button:**
    - Control ID: IDC_SHOW_DESIGNER
    - Type: CButton
    - Member: m_ShowDesigner

Adding member variables and functions to the CEmbeddable_Designer class

Add the following member variables, and member functions to the CEmbeddable_Designer class. The code for the functions will be added in "Step 3: Writing the code" on page 112.

1. Create the following member variables.
   - Application pointer for the Craxddrt Application Object:
     - Variable Type: IApplicationPtr
     - Variable Name: m_Application
   - Report pointer for the Craxddrt Report Object:
     - Variable Type: IReportPtr
     - Variable Name: m_Report
2. Create the following member function:
   - Function to create or open a report and initialize the controls on the Embeddable Designer form.
     - Function Type: void
     - Function Declaration: InitReport(BOOL bNew)
3. Add a member function for the BN_CLICKED message of each of the following buttons:
   - **New Report**: (Click to create a new report)
     - Member function name: OnNewReport
   - **Open Report**: (Click to open an existing report)
     - Member function name: OnOpenReport

- **Save Report**: (Click to save the current report)
  - Member function name: OnSaveReport
- **Preview**: (Click to Preview the current report)
  - Member function name: OnPreview
- **Design**: (Click to edit the current report)
  - Member function name: OnShowDesigner

Step 3: Writing the code

In this step, you will write the code for member functions added in "Step 2: Adding member variables and member functions" on page 110; you will also add directives and constants to the source and header files.

This step is broken into 2 sections:

- "Adding directives and constants to the source and header files" on page 112
- "Adding the code for the CEmbeddable_Designer class" on page 112

Adding directives and constants to the source and header files

Add the following directives and constants to the Source and Header files.

1 Add a constant to Embeddable_DesignerDlg.cpp.

```
// Filter for the Open dialog box and the Save As dialog box
static const char BASED_CODE szFilter[] = "Crystal Reports|*.rpt||";
```

2 Add the directive to Embeddable_Designer.h

```
//Import the Crystal Reports 8.5 ActiveX Designer Design and Runtime Library
#import "craxddrt.tlb" no_namespace
```

Note: Copy Craxddrt.tlb to your application directory. Craxddrt.tlb can be found in the `\Crystal Reports\Developer Files\include` path, off the directory you installed Crystal Reports to.

Adding the code for the CEmbeddable_Designer class

Type or insert the code for the member functions created in "Adding member variables and functions to the CEmbeddable_Designer class" on page 111. The CEmbeddable_DesignerDlg class handles the basic functionality of the Embeddable Designer sample application. The class will handle all of the report functionality including creating a new report or opening an existing report, and displaying it in the Embeddable Designer or the Crystal Report Viewer. Once a report is designed or edited it can be saved, previewed, or refreshed.

Note: Error handling is not included in the code samples.

1 Add the code to the OnInitDialog function.

```
 // Hide the Crystal Reports Viewer control and show the
    // Embeddable Designer oontrol.
    m_Viewer.ShowWindow(SW_HIDE);
    m_Designer.ShowWindow(SW_SHOW);
```

2 Add the code for the InitReport function.

```
// In this function a report is created or opened. The report is set to
// the Embeddable Designer and Crystal Report Viewer, with the Embeddable
// Designer visible and the Crystal Report Viewer hidden. Then the
// command buttons are initialized so that Save Report, and Preview are
// enabled, and Design is disabled and hidden.
void CEmbeddable_DesignerDlg::InitReport(BOOL bNew)
{
    // Create a new report for designing and previewing.
    if(bNew)
    {
        m_Application.CreateInstance("CrystalDesignRuntime.Application");
        m_Report = m_Application->NewReport();
    }

    // Show an Open dialog box, and then browse and open a report
    // for editing and previewing.
    else
    }
        // Initialize the Open dialog box to only display the
        // Crystal Report(.rpt) file type.
        CFileDialog FileDlg(TRUE,"*.rpt",NULL, OFN_EXPLORER, szFilter);

        // Show the Open dialog box
        int iReturn = FileDlg.DoModal();

        if(iReturn == IDOK)
        {
            // Get the name and path of the report file selected from the
            // Open dialog box.
            _bstr_t FileName(FileDlg.GetPathName().AllocSysString());

            // Create the Application Object.

m_Application.CreateInstance("CrystalDesignRuntime.Application");

            // Open the report selected from the Open dialog box.
            m_Report = m_Application->OpenReport(FileName);
        }
        else if (iReturn = IDCANCEL)
        {
            // Terminate the execution of the function if the
            // user cancels.
            return;
        }
    }
    // Set the report object to the Embeddable Designer.
    m_Designer.SetReportObject(m_Report);

    // Set the report souce for the Crystal Reports Viewer
    m_Viewer.SetReportSource(m_Report);

    // Set the width of the report page to the width of the viewing area
    m_Viewer.Zoom(1);

    // View the report in the Crystal Reports Viewer control.
    m_Viewer.ViewReport();

    // Check if the Save Report button is enabled.
    // If the button is disabled then enable the Save Report,
    // and Preview buttons. The Save Report button will always
    // be enabled once the first report is opened or created.
    BOOL bEnabled;
    bEnabled = m_DesignerSettings.IsWindowEnabled();

    if( bEnabled == FALSE)
    {
        m_SaveReport.EnableWindow(TRUE);
        m_Preview.EnableWindow(TRUE);
    }

    // Check if the Embeddable Designer is hidden.
    // If it is hidden then hide the Design button, show the
    // Preview button, hide the Crystal Reports Viewer, and
    // show the Embeddable Designer.
    if(!m_Designer.IsWindowVisible())
    {
        m_ShowDesigner.ShowWindow(SW_HIDE);
        m_Preview.ShowWindow(SW_SHOW);
        m_Viewer.ShowWindow(FALSE);
        m_Designer.ShowWindow(TRUE);
    }

}
```

3 Add the code for the OnNewReport function.

```
// In this function call the InitReport function and pass a
// value of True to create a new report. Passing a value of
// False opens an existing report.
void CEmbeddable_DesignerDlg::OnNewReport()
{
    InitReport(TRUE);
}
```

4 Add the code for the OnOpenReport function.

```
// In this function call the InitReport function and pass
// a value of False to open an existing report. Passing a
// value of True creates a new report.
```

```
void CEmbeddable_DesignerDlg::OnOpenReport()
{
    InitReport(FALSE);
}
```

5 Add the code for the OnSaveReport function.

```
// In this function open a Save As dialog box and save the current
// report to the selected path and name.
void CEmbeddable_DesignerDlg::OnSaveReport()
{

    // Initialize the Save As dialog box to only display the Crystal
    // Report (.rpt) file type.
    CFileDialog FileDlg(FALSE,"*.rpt",NULL,OFN_EXPLORER, szFilter);

    // Show the Save As dialog box.
    int iReturn = FileDlg.DoModal();

    if(iReturn == IDOK)
    {
        // Get the path and name selected in the Save As dialog box.
        _bstr_t FileName(FileDlg.GetPathName().AllocSysString());

        // Save the report to the path and name specified in the
        // Save As dialog box.
        m_Designer.SaveReport(FileName);
    }
    else if(iReturn == IDCANCEL)
    {
        // Terminate the execution of the function if the
        // user cancels.
        return;
    }
}
```

6 Add the code for the OnPreview function

```
// In this function, refresh the report without refreshing the data
// source, hide the Preview button and the Embeddable Designer, and
// then show the Design button and the Crystal Report Viewer.
void CEmbeddable_DesignerDlg::OnPreview()
{
    m_Viewer.RefreshEx(FALSE);
    m_Preview.ShowWindow(SW_HIDE);
    m_Designer.ShowWindow(SW_HIDE);
    m_ShowDesigner.ShowWindow(SW_SHOW);
    m_Viewer.ShowWindow(SW_SHOW);
}
```

7 Add the code for the OnShowDesigner function.

```
// In this function, hide the Design button and the Crystal Report
// Viewer, and show the Preview button and the Embeddable Designer.
void CEmbeddable_DesignerDlg::OnShowDesigner()
{
    m_ShowDesigner.ShowWindow(SW_HIDE);
    m_Viewer.ShowWindow(SW_HIDE);
    m_Preview.ShowWindow(SW_SHOW);
    m_Designer.ShowWindow(SW_SHOW);
}
```

Step 4: Running the Embeddable Designer application

In this step you will:

- create and design a new report in the Embeddable Designer
- view the report and any changes made to the report
- open and edit an existing report in the Embeddable Designer
- view the report and any changes made to the report
- save a report.

1 With the application running, click **New Report**.
An empty report will appear in the Embeddable Designer.

Note: The interface for the Embeddable Designer is the same one used for ActiveX Designer reports created in the Microsoft Visual Basic IDE with the Report Designer Component.

2 Design a new report to view.
If you are not familiar with the designer environment, refer to "Designing reports in the Embeddable Designer" on page 117 for a step-by-step procedure on creating a simple report off the Xtreme Sample Database ODBC data source.

3 Click **Preview** to view the report in the Crystal Report Viewer.

4 Click **Design** and make some changes to the report in the Embeddable Designer. Then click **Preview** to see the changes in the Crystal Report Viewer .

5 Click **Open.** In the Open dialog box, select one of the sample reports and click **Open** to view the report in the Embeddable Designer dialog.

6 Click **Preview** to view the report in the Crystal Report Viewer.

7 Click **Design** and make some changes to the report in the Embeddable Designer. Then click **Preview** to see the changes in the Crystal Report Viewer .

8 Click **Save Report**.

9 In the Save As dialog box, browse to the desired directory, type the name you want to save the report as, and then click **Save**.

Note: When you are designing a report the Preview button will be visible, and when you are previewing a report the Design button will be visible.

Designing reports in the Embeddable Designer

Designing reports in the Embeddable Designer is an easy, intuitive process. In this procedure, you will create a simple report off the Xtreme Sample Database ODBC data source, which installs with Crystal Reports. The report will include a text object, database fields, and a group.

The procedure contains the following steps:

- "Step 1: Adding a data source to the report" on page 117
- "Step 2: Adding fields and grouping to the report" on page 117

Step 1: Adding a data source to the report

In this step, you will use the Data Explorer to add the Customer table from the ODBC data source Xtreme Sample Database.

1 On the **Main Report** tab, right-click **Database Fields**, and then click **Add Database to Report**.

2 In the Data Explorer dialog box, expand **ODBC** and expand **Xtreme Sample Database**; then select **Customer**, click **Add**, and click **Close**.

3 In the Visual Linking Expert dialog box, click **OK**.
The data source is now added to the report.

Step 2: Adding fields and grouping to the report

In this step you will add a text object to the Report Header section, the Customer Name and Last Year's Sales Fields to the Details section, and a group based on the Region field.

1 On the **Main Report** tab, right-click in the white space to access the shortcut menu.

2 On the shortcut menu, point to **Insert**, and then click **Text Object**.
An object frame appears with the Arrow pointer.

3 Drag the object frame to the **Report Header** section and click to release the text object.

4 Double-click the text object to edit it, and then type: `Xtreme Sales Report by Region`.

5 Expand **Database Fields**, and then expand **Customer** to view the database fields.

6 Drag the **Customer Name** and **Last Year's Sales** fields into the **Details** section.
A column heading for each field is automatically placed in the Page Header section.

7 Click **Insert Group** on the toolbar.

8 In the Insert Group dialog box, select **Region**, and then click **OK.**

You are now ready to preview, export, print, or save the report. See *Crystal Reports Online Help* (Crw.chm) for more information on designing reports in the Crystal Reports environment.

Note: You can find the typical Crystal Report Designer commands (menu items and toolbar buttons) on the shortcut menu of the Embeddable Designer. The Embeddable Designer does not support the Chart Analyzer, Olap Grids, or Maps.

Migrating to the RDC from the OCX 10

This chapter illustrates the benefits of using the RDC for integrating reporting functionality into your Visual Basic applications. Learn how to migrate applications from the OCX to the RDC to take advantage of the latest features within Crystal Reports. Also included is an overview of the RDC, its major components, object model, and a description of its advanced features not available within the OCX.

Overview

Since it was first included in Microsoft Visual Basic, Crystal Reports has become the world standard for desktop reporting and design. It has kept pace with technological advancements by giving Visual Basic developers new ways to integrate reporting into database applications. For example, the Crystal ActiveX Control (OCX)—the tool most Visual Basic developers are familiar with—was first introduced in 1995 with Crystal Reports 4.5. In June 1998, the Report Designer Component (RDC) was launched. It's a revolutionary tool designed specifically for Visual Basic developers to create, view and modify reports within the Visual Basic Integrated Development Environment (IDE).

Note: Visit the Seagate Software Developer Zone web site at `http://www.seagatesoftware.com/products/dev_zone.`
Click Support to get links for finding documentation and knowledge base articles about integrating reporting in your applications.

Summary

The RDC represents the latest in ActiveX technology and provides the following advantages over the OCX:

- Integrates directly into the Visual Basic IDE.
- Allows you to create, view, and modify reports using Reports Experts and familiar Visual Basic code.
- Exposes all Print Engine features and provides the greatest number of events and objects to write code to.
- Performs better because it is a dual interface component with no wrapper around the Print Engine.
- Takes advantage of code completion features that are easy to use in the Visual Basic editor.
- Is fully compatible with Microsoft Visual Basic 5.0 and 6.0.

OCX

OCX is the development interface most Visual Basic developers are familiar with because it has been a part of Crystal Reports since 1995. The OCX is based on an older version of ActiveX technology. All of its properties and methods are accessed through a single control. This limits your control of a report because it exposes only a subset of the Crystal Report Print Engine's functionality.

In addition, because the OCX acts a wrapper around the Print Engine, it's less efficient when loading a report because it can't directly access the Print Engine.

Code Comparison between the OCX and RDC

The RDC is based on the current generation of Microsoft ActiveX technology. It's the method Visual Basic developers must use to take full advantage of the features within the Crystal Report Print Engine. Applications that are created using the OCX will not be able to use the latest Crystal Reports technology. If you're planning future releases or new applications, and you'd like to use the most powerful and flexible tool, you should consider using the RDC.

You can benefit from using the RDC by getting increased control over reports, such as:

- flexible formatting like passing text to a text object
- enhanced printer control
- creating report templates with unbound fields and then binding the fields to a data source (or one of several data sources) at runtime
- report Variables that enable you to maintain state even if a report section needs to fire multiple times for a given record
- the Create API which lets you create new objects, and even new reports, at runtime using code
- the latest Print Engine features such as mapping and multiple parameters
- and most importantly, the ability to create, view and modify reports inside the Visual Basic IDE.

OCX and RDC sample application comparison

Below are two applications which provide similar functionality—the first is created using the OCX, the second uses the RDC. The RDC example shows how to create a new application or convert an existing OCX application. With very few exceptions, you can duplicate any properties and methods set by the OCX using the RDC. Its properties and methods are very similar to the OCX, greatly reducing the time it takes you to learn the product.

The major differences between the two applications include:

- Setting the Crystal-related Project References and Components.
- Setting or accessing objects to get to the properties or methods needed for the report.
- The addition of the Crystal Report Viewer for viewing reports.

Sample Application General Description

A report with a subreport is created off the xtreme.mdb database.

- The main report contains the Customer table and a parameter field.
- The subreport contains the Orders table and a formula field.

The OCX application consists of a Form with three Command Buttons and the OCX control.

Form Load:

- The Report is opened.
- The location of the database in the main report is changed.
- The parameter in the main report is set.
- The subreport is opened.
- The location of the database in the subreport is changed.
- A string is passed to the formula field in the subreport.

Command1

The report is previewed to screen.

Command2

- The printer is selected.
- The report is printed.

Command3

- The export options are set to export the report to a Rich Text Format.
- The report is exported.

A second form will be added when the application is created using the RDC. The Crystal Report Viewer is added to the second form for viewing the report.

OCX sample application

Project|References:

No Crystal References required

Project|Components:

Crystal Report Control

Form1

```
Private Sub Form_Load()
```

Open the report.

```
CrystalReport1.ReportFileName = App.Path & "\OCX_to_RDC.rpt"
```

Change the location of the database.

```
CrystalReport1.DataFiles(0) = App.Path & "\xtreme.mdb"
```

Pass the parameter value to the main report.

```
CrystalReport1.ParameterFields(0) = "Param1;Main Report Param;True"
```

Pass the selection formula to the main report.

```
CrystalReport1.ReplaceSelectionFormula _
"{Customer.Last Year's Sales} < 50000.00"
```

Open the subreport.

```
CrystalReport1.SubreportToChange = "Sub1"
```

Change the location of the database in the subreport.

```
CrystalReport1.DataFiles(0) = App.Path & "\xtreme.mdb"
```

Pass the formula to the subreport.

```
CrystalReport1.Formulas(0) = "Formula1= " & "'Subreport Formula'"
```

Set CrystalReport1 back to using the main report.

```
CrystalReport1.SubreportToChange = ""
End Sub
Private Sub Command1_Click()
```

Set the destination to window.

```
CrystalReport1.Destination = crptToWindow
```

Preview the Report.

```
CrystalReport1.Action = 1
End Sub
Private Sub Command2_Click()
```

Set the printer driver.

```
CrystalReport1.PrinterDriver = “HPPCL5MS.DRV”
```

Set the printer port.

```
CrystalReport1.PrinterName = “HP LaserJet 4m Plus”
```

Set the printer name.

```
CrystalReport1.PrinterPort = “\\Vanprt\v1-1mpls-ts”
```

Set the destination to printer.

```
CrystalReport1.Destination = crptToPrinter
```

Print the report.

```
CrystalReport1.Action = 1
End Sub
Private Sub Command3_Click()
```

Set the Report to be exported to Rich Text Format.

```
CrystalReport1.PrintFileType = crptRTF
```

Set the Destination to Disk.

```
CrystalReport1.Destination = crptToFile
```

Set the path and name of the exported document.

```
CrystalReport1.PrintFileName = App.Path & "\OCXExport.rtf"
```

Export the report.

```
CrystalReport1.Action = 1
End Sub
```

RDC sample application

To migrate this application to the RDC, remove the OCX component from Form1, and remove the Crystal Report Control from the Project | Components menu, in addition to the steps below:

Project | References

Reference the Crystal Report 8 ActiveX Designer Runtime Library

Project | Components

Crystal Report Viewer Control

Add a second form

Add the Crystal Report Viewer Control to Form2

The properties and methods are accessed from individual objects. Following this code sample is a detailed description on the RDC Automation Servers Object Model.

The RDC will open a standard Crystal Report (.RPT). The Report could have been Imported into or recreated in the RDC ActiveX Designer (.DSR).

Form1:

Declare the application object used to open the rpt file.

```
Dim crxApplication As New CRAXDRT.Application
```

Declare the report object.

```
Public Report As CRAXDRT.Report
Private Sub Form_Load()
```

Declare a DatabaseTable object for setting the location of the database. This object will be used for the main and subreport.

```
Dim crxDatabaseTable As CRAXDRT.DatabaseTable
```

Declare a ParameterFieldDefinition object for passing parameters.

```
Dim crxParameterField As CRAXDRT.ParameterFieldDefinition
```

Declare a Report object to set to the subeport.

```
Dim crxSubreport As CRAXDRT.Report
```

Declare a FormulaFieldDefinition object for passing formulas.

```
Dim crxFormulaField As CRAXDRT.FormulaFieldDefinition
```

Open the report.

```
Set Report = crxApplication.OpenReport _
(App.Path & "\OCX_to_RDC.rpt", 1)
```

Use a For Each loop to change the location of each DatabaseTable in the Reports DatabaseTable Collection.

```
For Each crxDatabaseTable In Report.Database.Tables
     crxDatabaseTable.Location = App.Path & "\xtreme.mdb"
Next crxDatabaseTable
```

Set crxParameterField to the first parameter in the parameterfields collection of the main report.

```
Set crxParameterField = Report.ParameterFields.Item(1)
```

Pass the value to the main report.

```
crxParameterField.AddCurrentValue "Main Report Parameter"
```

Set crxSubreport to the subreport 'Sub1' of the main report. The subreport name needs to be known to use this method.

```
Set crxSubreport = Report.OpenSubreport("Sub1")
```

Use a For Each loop to change the location of each DatabaseTable in the Subreport Database Table Collection.

```
For Each crxDatabaseTable In
     crxSubreport.Database.Tables
         crxDatabaseTable.Location = App.Path &
         "\xtreme.mdb"
Next crxDatabaseTable
```

Set crxFormulaField to the first formula in the formulafields collection of the subreport.

```
Set crxFormulaField = crxSubreport.FormulaFields.Item(1)
```

Pass the formula to the subreport.

```
crxFormulaField.Text = "'Subreport Formula'"
End Sub
Private Sub Command1_Click()
```

Call Form2 to preview the Report.

```
Form2.Show
End Sub
Private Sub Command2_Click()
```

Select the printer for the report passing the Printer Driver, Printer Name and Printer Port.

```
Report.SelectPrinter “HPPCL5MS.DRV”, “HP LaserJet 4m Plus”, “\\Vanprt\v1-
1mpls-ts”
```

Print the Report without prompting the user.

```
Report.PrintOut False
End Sub
Private Sub Command3_Click()
```

Declare an ExportOptions Object.

```
Dim crxExportOptions As CRAXDRT.ExportOptions
```

Set crxExportOptions to the Report object's ExportOptions.

```
Set crxExportOptions = Report.ExportOptions
```

Set the report to be exported to Rich Text Format.

```
crxExportOptions.FormatType = crEFTRichText
```

Set the destination type to disk.

```
crxExportOptions.DestinationType = crEDTDiskFile
```

Set the path and name of the exported document.

```
crxExportOptions.DiskFileName = App.Path & "\RDCExport.rtf"
```

Export the report without prompting the user.

```
Report.Export False
End Sub
```

Form2:

```
Private Sub Form_Load()
```

Set the Report source for the Crystal Report Viewer to the Report.

```
CRViewer1.ReportSource = Form1.Report
'View the Report
CRViewer1.ViewReport
End Sub
Private Sub Form_Resize()
```

This code resizes the Report Viewer control to Form2's dimensions.

```
CRViewer1.Top = 0
CRViewer1.Left = 0
CRViewer1.Height = ScaleHeight
CRViewer1.Width = ScaleWidth
End Sub
```

Working with Visual C++ and Visual InterDev

11

This chapter illustrates how the Report Designer Component (RDC) can be integrated using other development languages. Included are examples in Visual C++ and Visual Interdev.

Overview

While much of this Developer's Guide is aimed at the Visual Basic developer, it is important to note that the integration methods can be used in any environment that supports COM. This chapter shows you how to use different integration methods with other popular development environments.

Note: Visit the Seagate Software Developer Zone web site at
`http://www.seagatesoftware.com/products/dev_zone`
Click Support to get links for finding documentation and knowledge base articles about the Report Designer Component.

Using the RDC with Visual C++

There are different ways to access the Report Designer Component (RDC) and any other COM Automation server through Visual C++. This section describes how to use the #import method.

Manipulating the RDC in Microsoft Visual C++ involves three steps:

- Define and instantiate a variable to be used to manipulate the RDC COM object.
- Instantiate an actual instance of the RDC COM object and assign it to a variable.
- Manipulate the Properties and Methods and then output the report.

The following section leads you through the details in each of these steps.

Printing a Report through Visual C++

To print a report through Visual C++

1. Open Visual C++ 6.0 if it isn't already running.
2. From the File menu, select New. In the New dialog box select the MFC AppWizard (exe) from the Projects tab. Type in MyRDC for the Project Name and click OK.
3. Click the Finish button to accept the defaults for the MFC AppWizard and then click **OK** in the New Project Information dialog box.
4. Add a reference to the RDC runtime object Model. From the File | Open menu select the StdAfx.h file that was generated by the MFC App wizard and Click **Open**. Add the following line after the #include directives:

```
#import "C:\Program Files\Seagate Software\Crystal Reports\Developer
Files\include\craxdrt.tlb"
```

Note: This is the default location of the RDC runtime object model (craxdrt.dll).

5 Before you can invoke an RDC object, you must initialize OLE. From the File | Open menu, open the MyRDC.CPP file and add the following code:

```
struct InitOle {
InitOle()  { ::CoInitialize(NULL); }
~InitOle() { ::CoUninitialize();   }
} _init_InitOle_;
```

6 For optional Variant parameters declare a dummy variable in the MyRDC2.CPP class.

```
Variant dummy;
```

7 Add the following constants in the declaration’s of the MyRDC2.CPP class:

```
// The constants needed to create the Application and Report Objects COM
objects
const CLSID CLSID_Application =
{0xb4741fd0,0x45a6,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};
const IID IID_IApplication =
{0x0bac5cf2,0x44c9,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};
const CLSID CLSID_ReportObjects =
{0xb4741e60,0x45a6,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};
const IID IID_IReportObjects =
{0x0bac59b2,0x44c9,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};
```

Opening and Printing a Crystal Report through the RDC

To open and print a Crystal Report through the RDC

1 From the File | Open menu open the MyRDC.CPP file. Add the following code to the MyRDC.InitInstance() method just before the return statement:

```
// A dummy variant
VariantInit (&dummy);
dummy.vt = VT_EMPTY;
HRESULT          hr = S_OK;
             IApplicationPtr m_Application = NULL;
             IReportPtr m_Report = NULL;

             // Specify the path to the report you want to print
_bstr_t ReportPath(“c:\\Program Files\\Seagate Software\\Crystal
Reports\\Samples\\En\\Reports\\General Business\\Inventory.rpt”);
_variant_t vtEmpty(DISP_E_PARAMNOTFOUND, VT_ERROR);
// Instantiate the IApplication object
hr = CoCreateInstance(CLSID_Application, NULL, CLSCTX_INPROC_SERVER ,
IID_IApplication, (void **) & CRAXDRT::IApplication);
//Open the Report using the OpenReport method
m_Report = m_Application->OpenReport(ReportPath, dummy)
//Print the Report to printer
m_Report->PrintOut(dummy, dummy, dummy, dummy
);
```

2 Finally, from the Build menu, select Rebuild All.

There are hundreds of properties, methods, and events in the Report Designer Component object model that you can manipulate through your code to meet the demands of virtually any reporting requirement.

Using the RDC with Visual InterDev

The information below is directed at the developer of Web applications using the Report Integration Controls.

Installation

When you install Crystal Reports, the Report Integration Controls are installed in the folder SEAGATE SOFTWARE\SHARED\DESIGN TIME CONTROL by default. Initially, to set up the Report Integration Controls as Design Time Controls (DTC's) for use in Visual InterDev 6.0, you need to:

1. Right-click in the Visual InterDev Toolbox. Choose **Customize Toolbox** from the shortcut menu.
2. In the Customize Toolbox dialog box, select the Design Time Controls tab. Check the two items ReportSource and ReportViewer.

   The Report Integration Controls are comprised of the two DTC's, ReportSource and ReportViewer. You will find that these two DTC's are added to the Visual InterDev Toolbox. We will call them Report Source Control and Report Viewer Control below.

Adding Reports to the Current Project

You must first add a report to the current Visual InterDev project before you can apply any DTC's to it.

Insert a Report Source Control (DTC)

When you insert a Report Source Control into your Web application, you actually create an instance of the Report Source Control. Its name is ReportSourcei, where i is the i-th instance of the Report Source Control, i=1,2,3…

When you right-click on an instance of a Report Source Control, you will invoke the Report Source Properties dialog box.

The properties you can specify for an instance of a Report Source Control include the following:

Attach a report to the Report Source Control

General information: In the General tab of the Report Source Properties dialog box, you may attach a report to this instance of the Report Source Control from the list of reports included in the current project.

Logon and test connectivity

Logon information for database: In the Accounts tab of the Report Source Properties dialog box, you can specify the server and database names for each of the tables used in the attached report, and test connectivity for each of these servers after entering the logon information.

If the report is hosted on a Web Reports Server, you may leave the logon information blank, which will in turn prompt the end user to enter the logon information. If the report is hosted on an ASP server, then you must enter the logon information.

Specify how to handle parameters

Parameters: In the Parameters tab of the Report Source Properties dialog box, if the report is hosted on a Web Reports Server, you may specify whether to prompt the end user to choose from a default list of values, or enter a value for a parameter.

However, if the report is hosted on an ASP server, then you must choose a value for each input parameter.

Specify selection formulas

Selection formula: In the Formula tab of the Report Source Properties dialog box, you may specify your own selections formulas, or to modify existing selection formulas.

Insert a Report Viewer control (DTC)

When you insert a Report Viewer Control into your Web application, you create an instance of the Report Viewer Control. Its name is ReportVieweri, where i is the i-th instance of the Report Viewer Control, i=1,2,3…

When you right-click on an instance of a Report Viewer Control, you will invoke the Report Viewer Properties dialog box.

Point the Viewer at the Report Source Control

The properties you can specify for an instance of a Report Viewer Control include:

General information: In the General tab of the Report Viewer Properties dialog box, you can select an instance of a Report Source Control and attach it to this instance of the Report Viewer Control. Upon clicking the Advanced button, you can specify a virtual path which is an alias to the directory of the Report Viewer Control.

Options for end users: In the Options tab of the Report Viewer Properties dialog box, you may specify the following for the end user:

- the Report Viewer to use, whether it is the Report Viewer for Java or Report Viewer for ActiveX
- the language to use for the Report Viewer, depending on the fonts installed on the current system
- the size of the Report Viewer

- the ability to refresh report in the Report Viewer
- the ability to print report in the Report Viewer
- the ability to export report from the Report Viewer
- the ability to search in report in the Report Viewer
- the ability to drill down in report in the Report Viewer
- the ability to prompt on refresh of report in the Report Viewer
- the ability to generate group tree in the Report Viewer
- the ability to display group tree in the Report Viewer.

When you click OK, you're finished.

Index