Crystal Reports

Migrating from the OCX Control to the Crystal Reports 9 Report Designer Component (RDC)

Overview

This document helps illustrate the benefits of using the Crystal Reports 9 Report Designer Component (RDC) for integrating reporting functionality into Visual Basic applications. It also explains how Visual Basic developers can migrate from the OCX (a component provided in previous versions of Crystal Reports) to the RDC as their primary integration method. This document includes an overview of the RDC, descriptions of its major components and object model, and an outline of its advanced features not available within the OCX.

Contents

INTRODUCTION ................................................................. 3

THE REPORT DESIGNER COMPONENT .......................... 3

RDC Architecture ......................................................... 3
Understanding the RDC Object Model .............................. 4

THE OCX ........................................................................... 4

CODE COMPARISON BETWEEN THE OCX AND RDC ... 4

OCX and RDC Sample Application Comparison ................. 5
OCX Version .................................................................... 5
RDC Version .................................................................... 8

FEATURES EXCLUSIVE TO THE RDC ........................ 11

Printer options .......................................................... 11
Set Paper Orientation and Paper Size ........................... 11
Set PaperSource and PrinterDuplex ............................... 11
Display a Windows Standard Printer Setup Dialog Box .... 12

Enhanced parameters ................................................. 12
Set multiple values for a parameter ............................... 12
Set a value range for a parameter ................................. 13

Report Viewer ........................................................... 13
Customize the Export Button ........................................ 13

Format reports at runtime ........................................... 14
DSR ........................................................................ 14
RPT ......................................................................... 15
Pass text to a TextObject .............................................. 15
Format a field at runtime .............................................. 16
Format a field in a section event ................................. 17
Load an image in a section event ................................ 18

Active Data ............................................................... 20
Report Creation at Runtime ................................................................. 21
Using the Report Expert ................................................................. 21
Using the Report Creation API ....................................................... 22
Unbound Fields ............................................................................. 23
Change the Runtime Location of an OLE Object............................ 25

SUMMARY ......................................................................................... 26

APPENDIX A: UNDERSTANDING THE RDC OBJECT MODEL .......... 27
Accessing Objects ........................................................................... 27
Setting a Property or Method for All Objects in a Collection .......... 29

APPENDIX B: GRAPHICAL OVERVIEW OF THE RDC OBJECT MODEL ....... 31

APPENDIX C: CODE COMPARISON: OCX TO RDC ......................... 32
Properties ....................................................................................... 32
Methods ......................................................................................... 69

APPENDIX D: FEATURES COMPARISON OF OCX TO RDC................. 78
Introduction

Since it was first included in Microsoft® Visual Basic®, Crystal Reports™ has been a market-leading reporting tool with millions of licenses shipped. It has kept pace with technological advancements by giving Visual Basic developers new ways to integrate reporting into database applications. The Crystal ActiveX® Control (OCX)—the tool many Visual Basic developers are familiar with—was first introduced in 1995 with Crystal Reports 4.5. In June 1998, we launched the Report Designer Component (RDC), a much more robust tool designed specifically for Visual Basic developers to create, view and modify reports within the Visual Basic Integrated Design Environment (IDE).

The purpose of this technical brief is to illustrate the benefits of using the newest version of the RDC technology—Version 9—for integrating reporting functionality into Visual Basic applications, and to help you migrate applications from the OCX to the RDC. It includes an overview of the RDC, descriptions of the major components and the object model, and an outline of the advanced features not available within the OCX.

Note: Crystal Reports 9 does not include the OCX component. Crystal Decisions will continue to focus its research and development efforts on the RDC and, for web applications, the new Report Application Server. Developers who have created applications using the OCX and plan to create new applications with the RDC will find this paper an ideal resource for reference.

The Report Designer Component

The Report Designer Component (RDC) is a powerful, integrated solution for Visual Basic developers to quickly and easily integrate reporting into their database applications. It is 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, so that developers can open, design and customize reports within the Visual Basic IDE. Intuitive Report Experts make it flexible and efficient for connecting to data and integrating powerful reports into applications. With hundreds of report properties, methods and events, developers have complete control over their report designs, using familiar Visual Basic code. Report distribution is simplified through a small component count and free runtime for desktop applications. Reports can thus be packaged within the application’s executable, or stored outside the application in the traditional (.rpt) format.

RDC Architecture

The RDC consists of three components, which together, enable developers to create and program, as well as preview, print, and export their reports:

- The Automation Server (craxdrt9.dll)—also known as the RDC runtime engine—is an extensive object model with hundreds of properties and methods that developers can use to program a report.
- The Report Viewer (crviewer9.dll) lets you preview reports on-screen for greater control and flexibility over the viewed report.
• The **Report Designer** was created specifically for Visual Basic developers. It is integrated tightly within Visual Basic 5.0 and 6.0, providing developers with a more intuitive way to create, view, and modify reports in the Visual Basic IDE. Using the Report Designer, developers can create reports within their Visual Basic project and take advantage of Visual Basic features such as Microsoft Visual Source Safe, which makes creating a report almost as easy as inserting a form.

**Understanding the RDC Object Model**

The RDC 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 it doesn’t specify how components are structured, but defines how they communicate with each other, the RDC can be used in any development environment that supports COM, such as Visual Basic, Visual C/C++®, Visual InterDev®, etc.

You should also be familiar with Object Model Hierarchy in order to access the correct object or collection. See Appendix A for more information of the RDC Object Model and **Appendix B** for a diagram.

**The OCX**

The OCX is the development interface with which many Visual Basic developers are familiar because it had been a part of Crystal Reports since 1995. However, it is based on older technology and therefore presents limitations for application integration. All of its properties and methods are accessed through a single control, limiting control of a report by exposing only a subset of the Crystal Report Print Engine’s functionality. In addition, because it acts as a wrapper around the Crystal Report Print Engine (also referred to as the Report Engine or Print Engine), the OCX is less efficient when loading a report because it can’t directly access the Report Engine.

**Code Comparison between the OCX and RDC**

The RDC is based on the current generation of Microsoft ActiveX technology. It is the method Visual Basic developers must use to take full advantage of the features in the Crystal Report Print Engine. Applications that are created using the OCX will not be able to use the latest powerful Crystal Reports technology. If you are planning future releases or new applications, and you want the most powerful and flexible tool, consider the RDC. Note: The RDC is intended for use in desktop applications, for web application the recommended component to use is the Report Application Server.

Visual Basic developers can benefit from using the RDC with increased control over reports: flexible formatting such as passing text to a TextObject; the latest Report Engine features such as mapping, multiple parameters, report creation at runtime, and support for unbound fields; and the ability to create, view and modify reports inside the Visual Basic IDE.

To view the code for each property or method of the OCX and its RDC equivalent, please view **Appendix C**.
OCX and RDC Sample Application Comparison

The following sample applications provide similar functionality—the first is created using the OCX, and the second using the RDC. The RDC example shows how to create a new application or convert an existing OCX application. The RDC, with few exceptions, can duplicate any properties and methods set by the OCX. Its properties and methods are very similar to the OCX, greatly reducing the learning curve for developers.

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 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 Report Viewer is added to the second form for viewing the report.

OCX Version

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 Version

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 9 ActiveX Designer Runtime Library

Project | Components
Crystal Report Viewer Control 9

Add a second form
Add the Report Viewer to Form2

The properties and methods are accessed from individual objects. Following this code sample is a detailed description of the RDC Automation Server Object Model.

The RDC will open a standard Crystal Report (.RPT) that has either been imported into, or recreated in the RDC ActiveX Designer (.DSR). Refer to Appendix D for details on opening a .DSR file using the RDC object model, the .ReportFileName property for OCX.

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
Dim crxDatabaseTable as craxdrt.DatabaseTable

'Declare a Report object to set to the subeport
Dim crxSubreport As CRAXDRT.Report

'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
crxDBDatabaseTable.ConnectionString("Database Name") = App.Path & "\xtreme.mdb"
Next crxDBDatabaseTable


'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 crxDBDatabaseTable In crxSubreport.Database.Tables
    crxDBDatabaseTable.ConnectionString("Database Name") = App.Path & "\xtreme.mdb"
Next crxDBDatabaseTable

'Pass the formula's text to the first formula field in the FormulaFields collection of the subreport. CrxSubreport.FormulaFields.Item(1).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"
Private Sub Command3_Click()

    'Set the report to be exported to Rich Text Format
    Report.ExportOptions.FormatType = crEFTRichText

    'Set the destination type to disk
    Report.ExportOptions.DestinationType = crEDTDiskFile

    'Set the path and name of the exported document
    "\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 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

End Sub
Features Exclusive to the RDC

The RDC offers many advanced features that are not available in the OCX including the following:

- **Printer options** - set paper orientation and paper size, set duplex printing options and paper source, display a Window standard Printer Setup dialog box.
- **Enhanced parameters** - set multiple values for a parameter, set a value range for a parameter
- **Report Viewer** - view multiple reports, customize the Export button.
- **Format at runtime** - pass text to a text object, format a field at runtime, format a field in a section event, load a picture in a section event.
- **Report creation at runtime** – design and format a report at runtime through a Report Expert or code (runtime license applies)
- **Unbound fields** – bind fields at runtime.
- **Change the runtime location of an OLE object** – change the location of any OLE object through the Section Format event.
- **Conditional formulas** – set conditional formatting formulas for most objects

**Printer options**

The RDC is able to set printer options such as paper orientation and paper size at runtime allowing for greater control and flexibility in the printing of reports.

**Set Paper Orientation and Paper Size**

`'Set the paperorientation
Report.PaperOrientation = crLandscape`

`'Set the papersize
Report.PaperSize = crPaper11x17`

The RDC is able to set the paper source and duplex printing options at runtime. A Windows standard printer setup dialog box is also available to allow the user to change the printer properties directly at runtime. These additional features offer even greater control and flexibility in the printing of reports.
Private Sub Form_Load()
    'Set the paper source to the lower bin
    Report.PaperSource = crPRBinLower
    'Set duplex printing to horizontal
    Report.PrinterDuplex = crPRDP Horizontal

    'Set the Report to the Report Viewer
    CRViewer1.ReportSource = Report
    'View the report
    CRViewer1.ViewReport
End Sub

Display a Windows Standard Printer Setup Dialog Box
' Set the Report object to the DSR
Dim Report As New CrystalReport1

Private Sub Form_Load()
    'Call the Printer Setup dialog box
    Report.PrinterSetup Me.hWnd

    'Set the Report to the Report Viewer
    CRViewer1.ReportSource = Report
    'View the report
    CRViewer1.ViewReport
End Sub

Enhanced parameters

Set multiple values for a parameter
Use this method to set multiple default values for the parameter field. When the user is prompted at runtime, a list of the default values set will be available for the parameter.

    'Set the default values to be displayed from the Parameter dialog
    'For the first parameter of the ParameterFields collection
    'SetNthDefaultValue will replace and add to the list of 'default parameters.
Report.ParameterFields.Item(1).SetNthDefaultValue 1, 5000
Report.ParameterFields.Item(1).SetNthDefaultValue 4, 20000

Set a value range for a parameter

Use this method to set a range value for the parameter field. Consider for example, the ranged currency parameter is created ({$Sales Range}). In the report’s selection formula, the ‘Last Year’s Sales’ Field is set equal to the parameter range ({$Customer.Last Year’s Sales} = {$Sales Range}). At runtime using the RDC, the range for the parameter field is set using the AddCurrentRange method of the ParameterFieldDefinition object.

' Set the start and end values of the parameter’s range.  
' for the first parameter in the ParameterFields collection
' The third parameter indicates whether the upper and/or lower bound of the range should be included

' A second range can be passed to the report.  
' The report will now select all Last Year’s Sales  
' In the ranges of 1000 - 10000 and 20000 - 25000

Report Viewer

The Report Viewer is an ActiveX control invoked by the application to present one or more reports to the end user. Users of the viewer can navigate and analyze a report, print it, export it to a variety of formats, and much more. The Report Viewer exposes events for every object in the control plus most elements in the report itself, allowing developers to customize the behavior of the viewer.

Customize the Export Button

In the following example, the Export Button click event is handled to either show a custom form or export to a set format.

To access the events:

1. Open the Code window
2. Select CRViewer1 from the Object list
3. Select the desired event. (ExportButtonClicked)
Private Sub CRViewer1_ExportButtonClicked(UseDefault As Boolean)
  'Set UseDefault to false to disable default Export dialog
  UseDefault = False

  'Display your own custom export form
  frmExport.Show

  '-or-

  'Export to set format

  'Set destination to disk
  Report.ExportOptions.DestinationType = crEDTDiskFile
  'Set format to rich text
  Report.ExportOptions.FormatType = crEFTRichText
  'Set file name and path

  'Export without prompting user
  Report.Export False
End Sub

Format reports at runtime

One of the most powerful features of the RDC is the ability to format objects at runtime. This includes such features as passing text to a TextObject, moving and suppressing individual fields, setting font characteristics for fields, loading images at runtime, and setting properties based on a field’s value. The RDC uses reports created in the Crystal Report .RPT format and the RDC ActiveX Designer .DSR format. The format used and how the DSR is initiated will determine formatting capabilities and how the code is written.

DSR

The DSR can load images at runtime and perform code in the Format Section events of the report. An example is reading the value of a Currency field at runtime—any value under a specified amount would cause the field’s background color to be set to red.

If the code for formatting an object is written outside of the DSR form, the method used to initiate the DSR will determine how the object is accessed. Declaring a variable as a New DSR will allow direct access to all the objects on a report. Setting a Report Object to a DSR means it can only be accessed from the section in which it resides, the same as if the object was in an RPT.
When an application is compiled, the DSR becomes part of the executable. If the DSR changes, the application will need to be recompiled and redistributed.

**RPT**

The RPT code is the same as a Report object set to a DSR. To access the Format Sections events, Section objects will need to be declared ‘WithEvents’ and set to the appropriate section.

*Declaring a variable as a New DSR:*

```
'General Declarations

'CrystalReport1 is the name of the DSR in the Designer folder of 'the Project menu (CrystalReport1.dsr)
Dim Report as New CrystalReport1
```

*Setting a Report Object to a DSR:*

```
'General Declarations
Dim Report as craxdrt.Report

Private Sub Form_Load()

'Set the generic Report object to the DSR
Set Report = New CrystalReport1

End Sub
```

**Pass text to a TextObject**

In this example, the Report Title is passed to a TextObject in the Report Header. This following code applies to an RPT or a Report object set to a DSR.

```
'Declare a TextObject to pass text to
Dim crxTextObject As CRAXDRT.TextObject
'Declare a generic object for searching
'through all the objects in the section
Dim crxObject As Object
'Search through each Report object in the Report Header Section
For Each crxObject In Report.Sections.Item("RH").ReportObjects

'Check if the object is a TextObject
```
If crxObject.Kind = crTextObject Then

' Set crTextObject to the Object if true
Set crxTextObject = crxObject
' Pass the text to the TextObject using the
' SetText method of the TextObject
crxTextObject.SetText "Report Title"

End If

Next crxObject

This code applies to passing text to a variable declared as a New DSR.

Report.Text1.SetText "Report Title"

**Format a field at runtime**

In this example, the first field in the Detail Section is resized and moved, and the font is set to bold. This code applies to passing text to an RPT or a Report object set to a DSR.

' Declare a FieldObject to format
Dim crxFieldObject As CRAXDRT.FieldObject

' Set crxFieldObject to the first report object in the
detail
' section
Set crxFieldObject = Report.Sections.Item("D").ReportObjects.Item(1)

' Set the width of the field
crxFieldObject.Width = 1000
' Move the field to the left
crxFieldObject.Left = 200
' Make the field bold
crxFieldObject.Font.Bold = True

This code applies to passing text to a variable declared as a New DSR.

' Set the width of the field
Report.Field1.Width = 1000
'Move the field to the left
Report.Field1.Left = 200
'Make the field bold

Format a field in a section event

The report has two fields from the Employee table in the Detail section. Field1 is the ‘Employee Name’ field, and Field2 is the ‘Last Years Sale’s’ field.

This code applies to Section Format Events in a DSR. Since the code is in the code section of the DSR, it does not matter how the DSR is initiated.

'Section 3 is the detail section
Private Sub Section3_Format(ByVal pFormattingInfo As Object)

'Set the background color of the field depending on the amount of 'sales
If Field2.Value < 20000 Then
    Field2.BackColor = vbRed
Else
    Field2.BackColor = vbGreen
End If

End Sub

This code applies to Section Format Events in an RPT. The code is more intensive in the RPT example because the objects on the report are not directly accessible, and must be declared and set in order to access the properties.

'General Declarations

'Declare a Report object
Dim crxReport As CRAXDRT.Report
'Declare an Application object
Dim crxApplication As New CRAXDRT.Application
'Declare a Section object with events
Dim WithEvents Section3 As CRAXDRT.Section

Private Sub Form_Load()
'Open the Report
Set crxReport = crxApplication.OpenReport("c:\Test.rpt")
'Set Section3 to the Detail section of the report
Set Section3 = Report.Sections(3)

'Set the Report to the Viewer
CRViewer1.ReportSource = crxReport
'Preview the Report
CRViewer1.ViewReport

End Sub

Private Sub Section3_Format(ByVal pFormattingInfo As Object)
'Declare a Field object
Dim crxField As CRAXDRT.FieldObject

'Set crxField to the second reportobject in the
'Detail Section. This is Field2 on the Report
Set crxField = crxReport.Sections("D").ReportObjects.Item(2)
'Set the background color of the field depending on the
'amount of 'sales
If crxField.Value < 20000 Then
    crxField.BackColor = vbRed
Else
    crxField.BackColor = vbGreen
End If

End Sub

Load an image in a section event

The report has three fields from the Employee table and an OLE Object—Object
Type set to Bitmap—in the Detail section. Field1 is the ‘Employee Name’ field,
Field2 is the ‘Last Years Sale’s’ field, and Field3 is the path to the bitmap of the
employee’s picture. Field3 is suppressed because only the field value is needed
to load the picture.

This code applies to Section Format Events in a DSR. Since the code is in the
code section of the DSR, it does not matter how the DSR is initiated.

'Section 3 is the detail section
Private Sub Section3_Format(ByVal pFormattingInfo As Object)

'Pass the path from field3 to the Visual Basic LoadPicture function.
The picture is then loaded into the Ole Object
Set Picture1.FormattedPicture = LoadPicture(Field3.Value)

End Sub

This code applies to Section Format Events in an RPT. The code is more intensive in the .RPT example because the objects on the Report are not directly accessible, and must be declared and set in order to access the properties.

'General Declarations

'Declare a Report object
Dim crxReport As CRAXDRT.Report
'Declare an Application object
Dim crxApplication As New CRAXDRT.Application
'Declare a Section object with events
Dim WithEvents Section3 As CRAXDRT.Section

Private Sub Form_Load()
'Open the Report
Set crxReport = crxApplication.OpenReport("c:\Test.rpt")

'Set Section3 to the Detail section of the report
Set Section3 = Report.Sections(3)

'Set the Report to the Viewer
CRViewer1.ReportSource = crxReport
'Preview the Report
CRViewer1.ViewReport

End Sub

Private Sub Section3_Format(ByVal pFormattingInfo As Object)
'Declare a Field object
Dim crxField As CRAXDRT.FieldObject
'Declare an OLE object for the picture
Dim crxPicture As CRAXDRT.OLEObject

'Set crxPicture to the Fourth reportobject in the
'Detail Section. This is Picture1 on the Report
Set crxPicture =
crxReport.Sections("D").ReportObjects.Item(4)

'Pass the value from field3 (path to the bitmap) to the
'Visual Basic LoadPicture function.
'The picture is then loaded into the Ole Object
Set crxPicture.FormattedPicture =
LoadPicture(crxField.Value)

End Sub

Active Data

One of the cornerstones of the RDC is the ability to report off runtime data, i.e. set the data source for the report at runtime. The data source can consist of any valid recordset created using ADO, RDO, DAO, or CDO. While this is not an exclusive feature of the RDC, it is an important feature designed to give the programmer greater flexibility and control over the SQL used to create the report. The processing, filtering and sorting of the data can be moved from the report to the server, greatly increasing the speed and efficiency of the report.

'Set the Report object to the DSR
Dim Report As New CrystalReport1
'Declare an ADO recordset object
Dim ADOrs As New ADODB.Recordset

Private Sub Form_Load()

'Create a Filtered Recordset to pass to the report
'The Select portion of the SQL must be the same as the
'structure of the data definition used in the report
ADOrs.Open "Select * From Customer Where Customer.‘Customer Name’ Like ‘C%’", "database=uid=pwd=dsn=xtreme sample data"

'Set the data source to the ADO recordset.
Report.Database.SetDataSource ADOrs
Set the Report to the Report Viewer
CRViewer1.ReportSource = Report

'View the report
CRViewer1.ViewReport

End Sub

Report Creation at Runtime
Developers can choose between a Report Expert or code to integrate runtime report creation capabilities into their applications.

Using the Report Expert
Now users can create their own reports at runtime by using an intuitive Report Expert to add fields, add groups, create summaries, filter and format.

In the following example the Customer table from xtreme.mdb is added to a new report. The Report Expert is displayed. Step through the intuitive interface to create, save and display the report.

'Declare a new instance of the application object
Dim crxApplication As New CRAXDRT.Application

'Declare a Report object. The report is generated at runtime
Dim crxReport As CRAXDRT.Report

Private Sub Form_Load()
'Declare a Report Wizard object
Dim crxWizard As New CrystalReportWizard.CRStandardWizard

'Create a new blank report

'Add the Customer table from xtreme.mdb

'Set crxReport to the Crystal Report Wizard

'Display the Crystal Report Wizard
crxWizard.DisplayReportWizard
End Sub

Using the Report Creation API

Crystal Reports Version 8 and higher exposes a Report Creation API interface from the RDC. This allows requests to be created and modified entirely within the application code. The new report creation API functions allow runtime creation of report objects including text, database fields, unbound fields, charts, specials, boxes, cross-tabs, blob fields, lines, pictures, summaries and subreport objects. These can either be added at runtime to an existing report created in the Visual Basic designer, or to a blank report. All properties that are normally available for each object at runtime are also available for these objects.

The following is a limited example of the full capabilities of the Report Creation API. In this example:

• Create the report
• Add the Customer Table from xtreme.mdb
• Add the “Customer Name” and “Last Year’s Sales” fields
• Create a group from the “Country” field
• Add the “Country” field to the group header
• Display the report

'Add a Report Viewer control to the form.
'Declare a new instance of the Application object
Dim crxApplication As New CRAXDRT.Application
'Declare a Report object. The report is generated at runtime
Dim crxReport As CRAXDRT.Report

Private Sub Form_Load()
'Declare a DatabaseFieldDefinition object
Dim crxField As CRAXDRT.DatabaseFieldDefinition

'Create a new blank report

'Add the Customer table from xtreme.mdb to the report
crxReport.Database.Tables.Add "C:\Program Files\Seagate Software\Crystal Reports\Samples\Databases\xtreme.mdb", "Customer"

'Add the "Customer Name" field to the report and set the Top, Left position
crxReport.Sections("D").AddFieldObject "{Customer.Customer Name}", 0, 0

'Add the "Last Year's Sale's" field to the report and set the Top, Left position

crxReport.Sections("D").AddFieldObject "{Customer.Last Year's Sales}", 2000, 0

'Set crxField to the "Country" field
Set crxField = crxReport.Database.Tables(1).Fields(13)

'Add a group based off the "Country" field
crxReport.AddGroup 0, crxField, crGCAnyValue, crAscendingOrder

'Add the Country field to the newly created Group Header
crxReport.Sections("GH").AddFieldObject "{Customer.Country}", 0, 0

'Set the Report to the Report Viewer
CRViewer1.ReportSource = Report

'View the report
CRViewer1.ViewReport

End Sub

NOTE

All Runtime Report Creation functionality requires licensing. For more information, please visit the Crystal Decisions web site at http://www.crystaldecisions.com

Unbound Fields

Unbound fields are fields of a specific data type that are placed on the report and set to a data source at runtime. There are two methods for setting the data source. The first method names the unbound field and then searches through the report’s data source for a matching field. The second method sets the “{Table.FieldName}” of the unbound field.

In both examples, two unbound fields are placed on the report. The first is a number field, and the second is a string field.

Set the Name of the Unbound Field:

'Set the Report object to the DSR
Dim Report As New CrystalReport1

Private Sub Form_Load()

'Add the Database and table to the report at runtime
'The Database and table can also be set at design time
Report.Database.Tables.Add "c:\Databases\xtreme.mdb", "Customer"

'Set the names for the number and string fields. The name is


'not case sensitive but can not contain any spaces, or separators.
Report.UnboundNumber1.Name = "customerid"
Report.UnboundString1.Name = "customername"

'Search through the report's tables and fields
'for a matching field. The name of the field may
'contain spaces or separators
Report.AutoSetUnboundFieldSource crBMTName

'Set the Report to the Report Viewer
CRViewer1.ReportSource = Report
'View the report
CRViewer1.ViewReport
End Sub

Set the "{Table.FieldName}" of the unbound field:
'Set the Report object to the DSR
Dim Report As New CrystalReport1

Private Sub Form_Load()
'Add the Database and table to the report at runtime
'The Database and table can also be set at design time
Report.Database.Tables.Add "c:\Databases\xtreme.mdb", "Customer"

'Set the table and field names for the number and string fields.
'The Table.FieldName is not case sensitive but must follow the same
'syntax, i.e. spaces, separators.
Report.UnboundNumber1.SetUnboundFieldSource "{Customer.Customer Id}"

'Set the Report to the Report Viewer
CRViewer1.ReportSource = Report
'View the report
CRViewer1.ViewReport
End Sub
Change the Runtime Location of an OLE Object

One of the top features of the RDC is the ability to load an image into an OLE object at runtime through the section format event. Also a recent addition to the RDC feature set, allows the changing of the location of an embedded OLE object such as a Microsoft Word document or Microsoft Excel spreadsheet.

In this example, two OLE objects have been placed in the detail section of the report. One is named “crxOLEObjXls” and will be passed a Microsoft Excel spreadsheet, the other is named “crxOLEObjDoc” and will be passed a Microsoft Word document. All code is done through the section format event of the DSR.

    Private Sub Section3_Format(ByVal pFormattingInfo As Object)
        'Set the location of crxOLEObjXls to a Microsoft Excel spreadsheet
        crxOLEObjXls.SetOleLocation App.Path & "]Excel1.xls"
        'Set the Height and width of crxOLEObjXls
        crxOLEObjXls.Height = 1800
        crxOLEObjXLS.Width = 5791

        'Set the location of crxOLEObjDoc to a Microsoft Word document
        crxOLEObjDoc.SetOleLocation App.Path & "\Word1.doc"
        'Set the Height and width of crxOLEObjDoc
        crxOLEObjDoc.Height = 322
        crxOLEObjDoc.Width = 8641
    End Sub
Summary

The Report Designer Component represents the latest in ActiveX technology and provides the following advantages over the OCX:

• It integrates directly into the Visual Basic IDE.
• It allows you to create, view, and modify reports using Reports Experts and familiar Visual Basic code.
• It exposes all Print Engine features and provides the greatest number of events and objects to which to write code.
• It gives better performance from its dual-interface component, with no wrapper around the Print Engine.
• It allows you to take advantage of code completion features that are easy to use in the Visual Basic editor.
• It is fully compatible with Microsoft Visual Basic 5.0 and 6.0.

For more information on the Report Designer Component and Crystal Reports 9, please visit http://www.crystaldecisions.com/products/crystalreports
Appendix A: Understanding the RDC Object Model

The RDC is a dual-interface object model based on Component Object Model (COM) technology. COM is a Microsoft technology that was introduced with OLE 2, the second version of Object Linking and Embedding. It has since evolved to be the foundation for much more, including ActiveX controls and Automation Servers.

COM objects work by exposing their functions through an interface. Most COM objects support more than one interface.

Automation refers to the process by which Automation Servers and Automation Controllers communicate. An Automation Server is an application or component that exposes its functionality to other applications. An Automation Controller is another application or development tool—such Visual Basic, Visual C++, or Delphi—that uses the functionality exposed by the Automation Server to perform a task.

A COM object is programmed by invoking methods through the interfaces exposed by the Automation Server. The OCX is a COM object that exposes a single interface with a limited number of properties and methods. The Automation Server for the RDC exposes many interfaces providing much greater control at runtime.

A good understanding of COM and Automation is needed to properly program the RDC Automation Server. The RDC Automation Server’s Object Model is a hierarchy of objects and collections. A collection contains a number of like objects. For example, the DatabaseTables Collection contains all the DatabaseTable objects used in a report. Each collection has the same three properties: Count, Item, and Parent. Count returns the number of objects in a collection. Item is a one-based array that returns the specified object contained in the Collection. Parent is a reference to the Parent object. Each object has its own set of properties and methods.

Accessing Objects

Each object or collection can be declared as a variable of that type and set to the desired object or collection in the report. The object or collection can also be accessed by stepping through the object model using the dot '.' Operator. Once the desired object or collection is reached or set, its corresponding properties or methods can be implemented.

Coding access to a particular property or method can take from one to several lines of code depending on how the object is accessed. For example, the Name property of the DatabaseTable object provides the name of the Table.

The following code will demonstrate three methods to get to this property.

Example 1 will set a variable for each individual object from the Report object to the DatabaseTable object to get the Name property.

Example 2 will only set the variable for the DatabaseTable object before getting the Name property.
Example 3 will get the Name property directly. Each method will reduce the amount of code used. Solid knowledge of Object Hierarchy will allow for cleaner code.

Example 1:

' Declare and set each corresponding object.
' The Report Object is set when the report is opened or instantiated.

'Declare the Database Object
Dim crxDatabase as craxdrt.Database
'Declare the DatabaseTables Collection
Dim crxDatabaseTables as craxdrt.DatabaseTables
'Declare the DatabaseTable Object
Dim crxDatabaseTable as craxdrt.DatabaseTable

'The Database property of Report returns the Database object to `crxDatabase
Set crxDatabase = Report.Database

'The Tables property of crxDatabase returns the DatabaseTables `collection to crxDatabaseTables
Set crxDatabaseTables = crxDatabase.Tables

'The Item property of crxDatabaseTables returns the first DatabaseTable in the collection to crxDatabaseTable
Set crxDatabaseTable = crxDatabaseTables.Item(1)

'Print out the table name
MsgBox crxDatabaseTable.Name

Example 2:

' Declare and set only the DatabaseTable Object

'Declare the DatabaseTable Object
Dim crxDatabaseTable as craxdrt.DatabaseTable

'Starting with the Report Object step through the Object Model to `return the first DatabaseTable in the DatabaseTables Collection `to crxDatabaseTable.
Set crxDatabaseTable = Report.Database.Tables.Item(1)

'Print out the table name

Example 3:

`MsgBox crxDB databaseTable.Name`

'Starting with the Report Object step through the Object Model to access the Name property of the first DatabaseTable in `the DatabaseTables Collection.`

`MsgBox Report.Database.Tables.Item(1).Name`

### Setting a Property or Method for All Objects in a Collection

A collection may contain many objects, and the properties or methods for each of these objects may need to be set at runtime. Continuing with the DatabaseTable object, the following examples will get the Name property for all objects in the DatabaseTables collection.

Example 1 will get each DatabaseTable object individually.

Example 2 will use the Count property of the DatabaseTables collection to create a For Loop and cycle through all DatabaseTable objects collected.

Example 3 will use the Visual Basic `For Each` command to access each DatabaseTable object in the DatabaseTables collection.

**Example 1:**

```
Dim crxDB databaseTable as craxdrt.DatabaseTable

'Set crxDB databaseTable to the first DatabaseTable in the DatabaseTables collection.
Set crxDB databaseTable = Report.Database.Tables.Item(1)
'Set crxDB databaseTable = Report.Database.Tables.Item(2)
'Set crxDB databaseTable = Report.Database.Tables.Item(3)
'Set crxDB databaseTable = Report.Database.Tables.Item(4)
'Set crxDB databaseTable = Report.Database.Tables.Item(5)

'Print out the table name
MsgBox crxDB databaseTable.Name
```

**Example 2:**

```
Dim crxDB databaseTable as craxdrt.DatabaseTable

'Set crxDB databaseTable to the second DatabaseTable in the DatabaseTables collection.
Set crxDB databaseTable = Report.Database.Tables.Item(2)
'Set crxDB databaseTable = Report.Database.Tables.Item(1)
'Set crxDB databaseTable = Report.Database.Tables.Item(3)
'Set crxDB databaseTable = Report.Database.Tables.Item(4)
'Set crxDB databaseTable = Report.Database.Tables.Item(5)

'Print out the table name
MsgBox crxDB databaseTable.Name
```

**Example 3:**

```
Dim crxDB databaseTable as craxdrt.DatabaseTable

'Set crxDB databaseTable to the third DatabaseTable in the DatabaseTables collection.
Set crxDB databaseTable = Report.Database.Tables.Item(3)
'Set crxDB databaseTable = Report.Database.Tables.Item(1)
'Set crxDB databaseTable = Report.Database.Tables.Item(2)
'Set crxDB databaseTable = Report.Database.Tables.Item(4)
'Set crxDB databaseTable = Report.Database.Tables.Item(5)

'Repeat for all DatabaseTables.
```

**Example 2:**

```
Dim crxDB databaseTable as craxdrt.DatabaseTable

'Set crxDB databaseTable to the fourth DatabaseTable in the DatabaseTables collection.
Set crxDB databaseTable = Report.Database.Tables.Item(4)
'Set crxDB databaseTable = Report.Database.Tables.Item(1)
'Set crxDB databaseTable = Report.Database.Tables.Item(2)
'Set crxDB databaseTable = Report.Database.Tables.Item(3)
'Set crxDB databaseTable = Report.Database.Tables.Item(5)

'Repeat for all DatabaseTables.
```
Dim nTable as Integer

For nTable = 1 to Report.Database.Tables.Count
    'Set crxDBaseTable to the n-th DatabaseTable in the
    'DatabaseTables collection.

        Set crxDBaseTable = Report.Database.Tables.Item(nTable)

    'Print out the table name
    MsgBox crxDBaseTable.Name
    Next nTable

Example3:

    'Set crxDBaseTable to each DatabaseTable in the
    'DatabaseTables collection.
    For Each crxDBaseTable in Report.Database.Tables
        'Print out the table name
        MsgBox crxDBaseTable.Name
        Next crxDBaseTable
Appendix B: Graphical Overview of the RDC Object Model
Appendix C: Code Comparison: OCX to RDC

The following illustrates code for each property or method of the OCX, followed by the RDC equivalent.

Some OCX properties will contain many parameters. The RDC equivalent will be an object with many properties and may result in several lines of code for the RDC versus one line of code for the OCX. Although you may have to write a few more lines of code, by using the RDC and its separate objects you benefit from:

• The flexibility to set individual or multiple properties
• The ability to access separate objects in the report, and
• Greater control over the format of the entire report.

Some OCX properties or methods are either not implemented in the RDC or do not have a corresponding property or method. Where this is the case, alternate code or an alternate method is suggested, or the OCX property is obsolete and no alternative is provided within the RDC.

All properties are listed in alphabetical order.

### Properties

**Action:**

**OCX:**
Triggers the printing of the report to screen, printer or export.

_CrystalReport1.Action = 1_

**RDC:**
The RDC prints to screen, printer, and export through three different methods. All are methods of the Report object.

View in Report Viewer:
Viewing of the report is done through the Report Viewer Control:


_CrystalReportViewer1.ViewReport_

Printer:
Print the report without prompting the user.

_Report.Printout = False_

Export:
Export the report without prompting the user.

_Report.Export = False_

**BoundReportFooter:**

**OCX:**
Indicates whether or not a footer is printed at the bottom of each page with a page number when printing a bound report.

_CrystalReport1.BoundReportFooter = True_

**RDC:**
Note on Bound Reports:
The RDC does not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible.

In Crystal Reports 9, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply). For more information, please see the section on Report Creation in this document. The Report Footer section of a report template can be suppressed through code.

```
    Report.Sections.Item("RF").Suppress = True
```

**BoundReportHeading:**

**OCX:**
Specifies a report title to be displayed at the top of the first page of a bound report
```
    CrystalReport1.BoundReportHeading = "Box Office Report"
```

**RDC:**

**Note on Bound Reports:**
The RDC does not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible.

In Crystal Reports 9, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply). For more information, please go to page 17.

The ReportTitle field can be set through code.
```
    Report.ReportTitle = "Box Office Report"
```

**Connect:**

**OCX:**
Logs on to an SQL server or an ODBC data source.
```
    CrystalReport1.Connect = 
    "DSN=Accounting;UID=734;PWD=bigboard;DSQ=Administration"
```

**RDC:**

**ConnectionProperties:**
This is a method of the DatabaseTable Object.
```
   'Connect the first table of the tables collection to the data source
   Dim dbProperties As CRAXDRT.ConnectionProperties
   Set dbProperties = 
   Report.Database.Tables.Item(1).ConnectionProperties
   dbProperties("DSN") = "Accounting"
   dbProperties("Database") = Administration
   dbProperties("User ID") = "734"
   dbProperties("Password") = "bigboard"
```

**CopiesToPrinter**

**OCX:**
Specifies the number of copies to be printed if you are printing to a printer, if the value you assign to Destination is a value of 1-Printer.
CrystalReport1.CopiesToPrinter = 3

**RDC:**
Optional second parameter of the Printout method.
This is a method of the Report Object.

object.PrintOut PromptUser, NumberOfCopies, Collated, 
StartPageNumber, StopPageNumber

'Print to printer, without prompting the user, prints 3 
copies.
'collate the copies, start printing on page 1, 
'stop printing on page 5
Report.PrintOut False, 3, True, 1, 5

**DataFiles**

**OCX:**
Specifies the location of the database files or tables used in the report.

CrystalReport1.DataFiles(0) = "c:\new\xtreme.mdb"

**RDC:**

ConnectionProperties:
This is a method of the DatabaseTable Object.

'Set the database location for the first table of the 
tables collection
Report.Database.Tables.Item(1).ConnectionProperties("Database Location") = "c:\new\xtreme.mdb"

**DataSource**

**OCX:**
Specifies the DataControl for a bound Report.
Set at design time only.

**RDC:**

Note on Bound Reports:
The RDC version 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16.

In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply). For more information, please go to page 17.

**Destination**

**OCX:**
Specifies the destination to which your report is to be printed (Window, Printer, File or Mail).

'Send the report to a window

 CrystalReport1.Destination = 0

**RDC:**
The RDC prints to screen, printer and export through three different methods. All are methods of the Report object.
Print to screen:
Viewing of the report is done through the Report Viewer Control:

```vbnet
CRViewer1.ReportSource = Report
CRViewer1.ViewReport
```

Printer:
Print the report without prompting the user.

```vbnet
Report.Printout False
```

Export:
Export the report without prompting the user.

```vbnet
Report.Export False
```

**DetailCopies**

**OCX:**
Specifies the number of copies of each record in the Details section that the program is to print.

```vbnet
CrystalReport1.DetailCopies = 3
```

**RDC:**
CopiesToPrint
This is a property of the Area Object.

```vbnet
'Print 3 copies of each detail line
Report.Areas.Item("D").CopiesToPrint = 3
```

**DialogParentHandle**

**OCX:**
Specifies the handle of the parent window. The program uses this handle to determine the window within which it centers any dialog boxes it displays (progress dialog boxes, parameter field prompt dialog boxes, etc.).

```vbnet
CrystalReport1.DialogParentHandle = Form1.Hwnd
```

**RDC:**
SetDialogParentWindow
This is a Method of the Report Object.

```vbnet
'Set the Dialog Parent window to Form1
Report.SetDialogParentWindow Form1.hWnd
```

**DiscardSavedData**

**OCX:**
If data is saved with the specified report, setting this property to 1 (True) discards the data.

```vbnet
CrystalReport1.DiscardSavedData = True
```

**RDC:**
DiscardSavedData
This is a property of the Report Object.

```vbnet
Report.DiscardSavedData
```

**EMailCCList**

**OCX:**
Specifies the "CC" list to which you want your e-mail message sent.

```vbnet
CrystalReport1.EMailCCList = "John Brown; Jane Doe"
```
RDC:
MailCcList
This is a property of the ExportOptions Object.

'Set names for 'CC' list
Report.ExportOptions.MailCcList = "John Brown; Jane Doe"

**EMailMessage**

OCX:
Specifies the string you want to appear as the body of your e-mail message.

CrystalReport1.EMailMessage = "The meeting is at 4:00"

RDC:
MailMessage
This is a property of the ExportOptions Object.

'Set Email message
Report.ExportOptions.MailMessage = "The meeting is at 4:00"

**EmailSubject**

OCX:
Specifies the subject line in your e-mail message.

CrystalReport1.EMailSubject = "Staff meeting"

RDC:
MailSubject
This is a property of the ExportOptions Object.

'Set Email subject
Report.ExportOptions.MailSubject = "Staff meeting"

**EMailToList**

OCX:
Specifies the "To" list to which you want your e-mail message directed.

CrystalReport1.EMailToList = "Jane Doe"

RDC:
MailToList
This is a property of the ExportOptions Object.

'Set names for 'To' list
Report.ExportOptions.MailToList = "Jane Doe"

**EMailVIMBCCList**

OCX:
Specifies the "Blind CC" list to which you want your e-mail message copied.

CrystalReport1.EMailVIMBCCList = "John Jacobs; Jane Doe"

RDC:
MailBccList
This is a property of the ExportOptions Object.
'Set names for 'BCC’ list

Report.ExportOptions.MailBccList = "John Jacobs; Jane Doe"

ExchangeFolder

OCX:
Specifies the Exchange path to export a file, when you want to export to Microsoft Exchange.

CrystalReport1.ExchangeFolder = "c:\Microsoft\Exchange\NewRpt.rpt"

RDC:
ExchangeFolderPath
This is a property of the ExportOptions Object.

'Set the path for the exchange folder

Report.ExportOptions.ExchangeFolderPath = "c:\Microsoft\Exchange\NewRpt.rpt"

ExchangeProfile

OCX:
Specifies the Exchange Profile when you want to export to Microsoft Exchange.

CrystalReport1.ExchangeProfile = "James Andrews"

RDC:
ExchangeProfile
This is a property of the ExportOptions Object.

'Set the exchange profile


ExchangePassword

OCX:
Specifies the Exchange password when you want to export to Microsoft Exchange.

CrystalReport1.ExchangePassword = "pickle"

RDC:
ExchangePassword
This is a property of the ExportOptions Object.

'Set the exchange password

Report.ExportOptions.ExchangePassword = "Pickle"

Formulas

OCX:
Specifies a new string for an existing formula.

CrystalReport1.Formulas(0) = "COMMISSION= {file.SALES} * .1"
CrystalReport1.Formulas(1) = "TOTAL= {file.SALES} + {file.COMMISSION}"

RDC:
Text
This is a property of the FormulaFieldDefinition Object.

'Pass the formula to the first formula in the
'FormulaFields collection
Report.FormulaFields.Item(1).Text = "file.SALES} *.1"

'Pass the formula to the second formula in the
'FormulaFields collection
Report.FormulaFields.Item(2).Text = "{file.SALES} +
{file.COMMISSION}"
'Set the graph direction to Horizontal
Report.Sections.Item("GH1").ReportObjects.Item(1).GraphDirection = crHorizontalGraph

'Show no values on the label risers
Report.Sections.Item("GH1").ReportObjects.Item(1).DataPoint = crNone

'Show no gridlines for the Data Axis Grid Line

'Show Minor gridlines for the Group Axis Grid Line

'Show Major gridlines for the Series Axis Grid Line

'Show the graph legend
Report.Sections.Item("GH1").ReportObjects.Item(1).EnableShowLegend = True

'Set the maximum value to 5000

'Set the minimum value to 100
Report.Sections.Item("GH1").ReportObjects.Item(1).MinDataAxisValue = 100

---

**GraphText**

**OCX:**
Sets the various text components for the specified graph.

[form.]Report.GraphText(ArrayIndex%)={ sectionCode; graphNum;title; subTitle; footnote;series;group;x;y;z}$

CrystalReport1.GraphText(0) = "TITLE; 1; title string;
subtitle string; footnote string; series string; group string; x string;
y string; z string"

**RDC**
With the noted exceptions the parameters of the GraphText property are individual properties of the GraphObject. See below for the applicable GraphText parameter to GraphObject property match.
<table>
<thead>
<tr>
<th>OCX GraphText Parameter</th>
<th>RDC Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>SectionCode</td>
<td>Specify the item in the Sections collection that holds the graph: Report.Sections.Item(&quot;GH&quot;)</td>
</tr>
<tr>
<td>GraphNum</td>
<td>Specify the item in the ReportObjects collection of the Section the graph is in. Report.Sections.Item(&quot;GH&quot;).ReportObjects.Item(1)</td>
</tr>
<tr>
<td>Title</td>
<td>Title</td>
</tr>
<tr>
<td>SubTitle</td>
<td>SubTitle</td>
</tr>
<tr>
<td>Footnote</td>
<td>FootNote</td>
</tr>
<tr>
<td>Series</td>
<td>SeriesTitle</td>
</tr>
<tr>
<td>Group</td>
<td>GroupsTitle</td>
</tr>
<tr>
<td>X</td>
<td>XAxisTitle</td>
</tr>
<tr>
<td>Y</td>
<td>YAxisTitle</td>
</tr>
<tr>
<td>Z</td>
<td>ZAxisTitle</td>
</tr>
</tbody>
</table>

'The Graph is the first Object in the Group Header 1 section

'Set the Graph Title
Report.Sections.Item("GH1").ReportObjects.Item(1).Title = "Title String"

'Set the Graph SubTitle
Report.Sections.Item("GH1").ReportObjects.Item(1).SubTitle = "Subtitle String"

'Set the Graph Footnote
Report.Sections.Item("GH1").ReportObjects.Item(1).FootNote = "Footnote String"

'Set the Graph Series Title
Report.Sections.Item("GH1").ReportObjects.Item(1).SeriesTitle = "Series Title String"

'Set the Graph Groups Title
Report.Sections.Item("GH1").ReportObjects.Item(1).GroupsTitle = "Groups Title String"

'Set the Graph XAxis Title
Report.Sections.Item("GH1").ReportObjects.Item(1).XAxisTitle = "XAxis Title String"

'Set the Graph YAxis Title
Report.Sections.Item("GH1").ReportObjects.Item(1).YAxisTitle = "YAxis Title String"

'Set the Graph ZAxis Title
Report.Sections.Item("GH1").ReportObjects.Item(1).ZAxisTitle = "ZAxis Title String"

**GraphType**

**OCX:**
Sets the kind of graph used in the selected section in the specified report.

[form.]Report.GraphType(ArrayIndex%)=[sectionCode;graphNum;graphType$]

    CrystalReport1.GraphType(0) = "GROUPHDR.0.0;0; PIE"

**RDC**
With the noted exceptions the parameters of the GraphType property are individual properties of the GraphObject. See below for the applicable GraphType parameter to GraphObject property match.

<table>
<thead>
<tr>
<th>OCX GraphText Parameter</th>
<th>RDC Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>sectionCode</td>
<td>Specify the item in the Sections collection that holds the graph: Report.Sections.Item(&quot;GH&quot;)</td>
</tr>
<tr>
<td>graphNum</td>
<td>Specify the item in the ReportObjects collection of the Section the graph is in. Report.Sections.Item(&quot;GH&quot;).ReportObjects.Item(1)</td>
</tr>
<tr>
<td>graphType</td>
<td>GraphType</td>
</tr>
</tbody>
</table>

'The Graph is the first Object in the Group Header 1 section

'Set the GraphType to Multiple Pie
Report.Sections.Item("GH1").GraphType = crMultiplePieGraph

**GroupCondition**

**OCX:**
Specifies what kind of change in the Group Condition Field will trigger the creation of a group.

[form.]Report.GroupCondition(SequentialIndex%)=[ group; field; condition; sortDirection$]

    CrystalReport1.GroupCondition(0) = "GROUP1;{order details.ORDER ID};ANYCHANGE;A"

**RDC**
With the noted exceptions the parameters of the GroupCondition property are individual properties of the Area Object. See below for the applicable GroupCondition parameter to Area Object property match.

<table>
<thead>
<tr>
<th>OCX GroupCondition Parameter</th>
<th>RDC Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>group</td>
<td>Specify the item in the Areas collection that holds the</td>
</tr>
</tbody>
</table>
group: Report.Areas.Item("GH1")

<table>
<thead>
<tr>
<th>field</th>
<th>GroupConditionField</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>GroupCondition</td>
</tr>
<tr>
<td>sortDirection</td>
<td>SortDirection</td>
</tr>
</tbody>
</table>

'Declare a DatabaseFieldDefinition object
'The GroupConditionField property must be passed
'the field to group on and not a string
Dim crxDBField As CRAXDRT.DatabaseFieldDefinition

'Set crxDBField to the 12th field in the first table (OrderDate 'Field).
Set crxDBField = Report.Database.Tables.Item(1).Fields.Item(12)

'Change the field grouped on to the OrderDate Field in Group Header 1
Report.Areas.Item("GH1").GroupConditionField = crxDBField

'Change the condition the report groups on to biweekly in Group Header 1
Report.Areas.Item("GH1").GroupCondition = crGCBiweekly

'Change the Sort Direction to Ascending in Group Header 1
Report.Areas.Item("GH1").SortDirection = crAscendingOrder

GroupSelectionFormula

OCX:
Specifies the groups to be used when printing the report.
CrystalReport1.GroupSelectionFormula = "Sum ({order details.ORDER AMOUNT}, {customer.CUSTOMER ID}) < $10000"

RDC
GroupSelectionFormula
This is a property of the Report Object.
Report.GroupSelectionFormula = "Sum ({order details.ORDER AMOUNT}, {customer.CUSTOMER ID}) < $10000"

GroupSortFields

OCX:
Specifies the group field(s) that are to be used to sort your data when the report is printed.
CrystalReport1.GroupSortFields(0) = "-Count
((customer.CUSTOMER ID),{customer.REGION})"

RDC
Add
This is a property of the Report Object GroupSortFields collection.

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)".

```
' Declare a SummaryFieldDefinition Object
Dim crxSummaryField As CRAXDRT.SummaryFieldDefinition

' Getting the first Summary Field which is the "SUM of Customer.Last Year's Sales"
Set crxSummaryField = Report.SummaryFields.Item(1)

' Add the Group Sort Field
Report.GroupSortFields.Add crxSummaryField, crDescendingOrder
```

**LastErrorNumber**

**OCX:**
Returns the error code for the last runtime error.

```
' If error occurs, go to Error Handler
On Error GoTo ErrorHandler

' OCX Code

ErrorHandler:
    MsgBox CrystalReport1.LastErrorNumber
```

**RDC**
Use Visual Basic error handling.

```
' If error occurs, go to Error Handler
On Error GoTo ErrorHandler

' RDC Code

ErrorHandler:
    MsgBox Err.Number
```

**LastErrorString**

**OCX:**
Returns the error string for the last runtime error.

```
' If error occurs, go to Error Handler
On Error GoTo ErrorHandler

' OCX Code

ErrorHandler:
    MsgBox Err.Number
```
MsgBox CrystalReport1. LastErrorString

**RDC**

Use Visual Basic error handling.

'If error occurs, go to Error Handler

On Error GoTo ErrorHandler

'RDC Code

ErrorHandler:
    MsgBox Err.Description

---

**LogOnInfo**

**OCX:**

CrystalReport1.LogOnInfo[0] = "DSN = Accounting;UID = 734;PWD = bigboard;DSQ = Administration"

**RDC**

ConnectionProperties

This is a method of the DatabaseTable Object.

'Connect the first table of the tables collection to the data source

Dim dbProperties As CRAXDRT.ConnectionProperties
Set dbProperties = Report.Database.Tables.Item(1).ConnectionProperties
dbProperties("DSN") = "Accounting"
dbProperties("Database") = Administration
dbProperties("User ID") = "734"
dbProperties("Password") = "bigboard"

---

**MarginBottom**

**OCX:**

Sets the bottom margin for the specified report.

CrystalReport1.MarginBottom = 720

**RDC**

BottomMargin

This is a property of the Report Object.

Report.BottomMargin = 720

---

**MarginLeft**

**OCX:**

Sets the left margin for the specified report.

CrystalReport1.MarginLeft = 1440

**RDC**

LeftMargin
This is a property of the Report Object.

\[ \text{Report.LeftMargin} = 1440 \]

**MarginRight**

**OCX:**
Sets the right margin for the specified report.

\[ \text{CrystalReport1.MarginRight} = 1440 \]

**RDC**

RightMargin
This is a property of the Report Object.

\[ \text{Report.RightMargin} = 1440 \]

**MarginTop**

**OCX:**
Sets the top margin for the specified report.

\[ \text{CrystalReport1.MarginTop} = 720 \]

**RDC**

TopMargin
This is a property of the Report Object.

\[ \text{Report.TopMargin} = 720 \]

**ParameterFields**

**OCX:**
Changes the default value of the specified parameter field. When the prompting dialog box appears for the parameter field, the value you specify with this property is the value you are prompted with.

\[ \text{CrystalReport1.ParameterFields(0)} = \text{"parameter1;red;TRUE"} \]

**RDC**

AddCurrentValue
These are methods of the ParameterFieldDefinition Object. Access the individual parameter through the ParameterFieldDefinitions collection. The RDC takes advantage of the enhanced parameter fields of Seagate Crystal Reports 7 and is able to pass multiple, single and ranged values.

'Note the RDC has the ability to add multiple values to a single parameter.

'Add the value to the first parameter in the ParameterFields collection

\[ \begin{align*}
\text{Report.ParameterFields.Item(1).AddCurrentValue} & \text{ 1000} \\
\text{Report.ParameterFields.Item(1).AddCurrentValue} & \text{ 5000} \\
\text{Report.ParameterFields.Item(1).AddCurrentValue} & \text{ 10000}
\end{align*} \]

**Password**

**OCX:**
Enters the password needed to use database tables on a restricted Access .MDB file.

\[ \text{CrystalReport1.Password} = \text{"dogsncats"} \]

**RDC:**

ConnectionProperties
This is the second parameter in the SetSessionInfo method of the DatabaseTable object.

```vbcn```
'Set the session info for the first table in the
'DatabaseTables collection
Report.Database.Tables.Item(1).ConnectionProperties("Password") = "bigboard"
```

**PrintDay**

**OCX:**
Sets the day component of the print date (if different from the actual date the report is printed).

```vbcn```
CrystalReport1.PrintDay = 23
```

**RDC:**
PrintDate
This is a property of the Report Object. PrintDate sets the entire Print Date.

```vbcn```
Report.PrintDate = 8 / 27 / 99
```

**PrinterCollation**

**OCX:**
If you specify more than one copy to be printed (using the PrinterCopies property), PrinterCollation specifies whether or not the copies will be collated.

```vbcn```
CrystalReport1.PrinterCollation = 1
```

**RDC:**
Optional Third parameter of the Printout method.
This is a method of the Report Object.

```vbcn```
object.PrintOut PromptUser, NumberOfCopies, Collated, StartPageNumber, StopPageNumber
```

```vbcn```
'Print to printer, without prompting the user, prints 3 copies.
'collate the copies, start printing on page 1,
'stop printing on page 5
Report.PrintOut False, 3, True, 1, 5
```

**PrinterCopies**

**OCX:**
Sets the number of report copies to be printed.

```vbcn```
CrystalReport1.PrinterCopies = 3
```

**RDC:**
Optional parameter of the Printout method.
This is a method of the Report Object.

```vbcn```
object.PrintOut PromptUser, NumberOfCopies, Collated, StartPageNumber, StopPageNumber
```

```vbcn```
'Print to printer, without prompting the user, prints 3 copies.
'collate the copies, start printing on page 1,
'stop printing on page 5
Report.PrintOut False, 3, True, 1, 5

PrinterDriver

OCX:
Sets the name of the printer driver that is to print the report.

CrystalReport1.PrinterDriver = "Epson24.drv"

RDC:
First parameter of the Select Printer method.
This is a method of the Report Object.

object.SelectPrinter DriverName, PrinterName, PortName

Report.SelectPrinter ""Epson24.drv"", "Epson LQ-850", "LPT1"

PrinterName

OCX:
Sets the name of the printer that is to print the report.

CrystalReport1.PrinterName = "Epson LQ-850"

RDC:
Second parameter of the Select Printer method.
This is a method of the Report Object.

object.SelectPrinter DriverName, PrinterName, PortName

Report.SelectPrinter ""Epson24.drv"", "Epson LQ-850", "LPT1"

PrinterPort

OCX:
Sets the name of the printer port for the specified printer.

CrystalReport1.PrinterPort= "LPT1"

RDC:
Third parameter of the Select Printer method.
This is a method of the Report Object.

object.SelectPrinter DriverName, PrinterName, PortName

Report.SelectPrinter ""Epson24.drv"", "Epson LQ-850", "LPT1"

PrinterStartPage

OCX:
Sets the first page to be printed.

CrystalReport1.PrinterStartPage = 7

RDC:
Optional Fourth parameter of the Printout method.
This is a method of the Report Object.

object.PrintOut PromptUser, NumberOfCopies, Collated,
StartPageNumber, StopPageNumber

 'Print to printer, without prompting the user, prints 3
copies.  
'collate the copies, start printing on page 1,  
'stop printing on page 5  
Report.PrintOut False, 3, True, 1, 5

**PrinterStopPage**

**OCX:**
Sets the last page to be printed.

CrystalReport1.PrinterStopPage = 12

**RDC:**
Optional Fifth parameter of the Printout method.
This is a method of the Report Object.

object.PrintOut PromptUser, NumberOfCopies, Collated,
StartPageNumber, StopPageNumber

 'Print to printer, without prompting the user, prints 3
copies.  
'collate the copies, start printing on page 1,  
'stop printing on page 5  
Report.PrintOut False, 3, True, 1, 5

**PrintFileCharSepQuote**

**OCX:**
Sets the quote character used to enclose alphanumeric field data when printing to a file using character-separated
format.

CrystalReport1.PrintFileCharSepQuote = """

**RDC:**
CharStringDelimiter
This is a property of the ExportOptions Object.

'Set the character to separate strings
Report.ExportOptions.CharStringDelimiter = ",,"

**PrintFileCharSepSeparator**

**OCX:**
Sets the character(s) you want to use to separate the fields when printing to a file using the Character Separated
Value format.

CrystalReport1.PrintFileCharSepSeparator= "@"

**RDC:**
CharFieldDelimiter
This is a property of the ExportOptions Object.
'Set the character to separate fields
Report.ExportOptions.CharFieldDelimiter = "@"

PrintFileLinesPerPage

OCX:
Indicates the number of lines to be printed before the page break. The default is 60 lines.
    CrystalReport1.PrintFileCharSepSeparator = "@"

RDC:
NumberOfLinesPerPage
This is a property of the ExportOptions Object.
    'Set the number of lines per page
    Report.ExportOptions.NumberOfLinesPerPage = 50

PrintFileName

OCX:
Specifies the name of the file to which the report is to be printed.
    CrystalReport1.PrintFileName = "c:\crw\cust_rpt.txt"

RDC:
DiskFileName
This is a property of the ExportOptions Object.
    'Set the path and name of the exported file
    Report.ExportOptions.DiskFileName = "c:\crw\cust_rpt.txt"

PrintFileODBCPassword

OCX:
Used to export in ODBC format, this specifies the password that you need to connect to the data source.
    CrystalReport1.PrintFileODBCPassword = "merry%%5"

RDC:
ODBCDataSourcePassword
This is a property of the ExportOptions Object.
    'Set password to the ODBC data source
    Report.ExportOptions.ODBCDataSourcePassword = "merry%%5"

PrintFileODBCSource

OCX:
Used whenever you export in ODBC format. Specifies the name of the data source to which you want to export.
    CrystalReport1.PrintFileODBCSource = "pickle"

RDC:
ODBCDataSourceName
This is a property of the ExportOptions Object.
    'Set the ODBC data source name
    Report.ExportOptions.ODBCDataSourceName = "pickle"

PrintFileODBCTable
OCX:
Used to export in ODBC format, this specifies the name of the table to which you want to export in the data source.

    CrystalReport1.PrintFileODBCTable = "Employees"

RDC:
ODBCExportTableName
This is a property of the ExportOptions Object.

    'Set the ODBC data source table name
    Report.ExportOptions.ODBCExportTableName = "Employees"

PrintFileODBCUser

OCX:
Used to export in ODBC format, this specifies the User ID that you need to connect to the data source.

    CrystalReport1.PrintFileODBCUser = "LisaB"

RDC:
ODBCDataSourceUserID
This is a property of the ExportOptions Object.

    'Set the User ID for the ODBC data source
    Report.ExportOptions.ODBCDataSourceUserID = "LisaB"

PrintFileType

OCX:
Specifies the type of print file used when printing a report to a file.

    'Prints the report to a file in a tab separated format.
    CrystalReport1.PrintFileType = 1

RDC:
FormatType
This is a property of the ExportOptions Object.

    'Set the format type to tab separated values
    Report.ExportOptions.FormatType = crEFTTabSeparatedValues

PrintFileUseRptDateFmt

OCX:
When you are printing to a file, this indicates whether or not the program should save dates in the same date format (MMDDYY,DDMMYY etc.) that is used in the report, or optimize the dates for the file format you have selected.

    'Specifies that the program should print dates in the same format as used in the report.
    CrystalReport1.PrintFileUseRptDateFmt = 1

RDC:
UseReportDateFormat
This is a property of the ExportOptions Object.

    'Set the date format to be the same as report
    Report.ExportOptions.UseReportDateFormat = True

PrintFileUseRptNumberFmt
**OCX:**
When you are printing to a file, this indicates whether or not the program should print numbers in the same format (decimal places, negatives, etc.) that you used in the report, optimize the numbers for the file format you have selected.

```
'Specifies that the program should print numbers in the same formats as used in the report.
CrystalReport1.PrintFileUseRptNumberFmt = 1
```

**RDC:**
UseReportNumberFormat
This is a property of the ExportOptions Object.

```
'Set the number format to be the same as report
Report.ExportOptions.UseReportNumberFormat = True
```

**PrintMonth**

**OCX:**
Sets the month component of the print date (if different from the actual date the report is printed).

```
CrystalReport1.PrintMonth = 7
```

**RDC:**
PrintDate
This is a property of the Report Object. PrintDate sets the entire Print Date.

```
Report.PrintDate = 8 / 27 / 99
```

**PrintYear**

**OCX:**
Sets the year component of the print date (if different from the actual date the report is printed).

```
CrystalReport1.PrintYear = 1994
```

**RDC:**
PrintDate
This is a property of the Report Object. PrintDate sets the entire Print Date.

```
Report.PrintDate = 8 / 27 / 99
```

**ProgressDialog**

**OCX:**
Enables/disables the display of the Progress dialog box. The Progress dialog box displays the progress of the report when it is running (records read, records selected, and so forth).

```
CrystalReport1ProgressDialog = False
```

**RDC:**
DisplayProgressDialog
This is a property of the Report Object.

```
Report.DisplayProgressDialog = False
```

**RecordsPrinted**

**OCX:**
Determines the number of records actually printed.

```
Printed& = CrystalReport1.RecordsPrinted
```
RDC: NumberOfRecordPrinted
This is a property of the PrintingStatus Object.

`PREVIEW THE REPORT`
`CRViewer1.ReportSource = Report`
`CRViewer1.ViewReport`

`'PASS THE NUMBER OF RECORDS PRINTED TO THE PRINTED VARIABLE`
`Printed& = Report.PrintingStatus.NumberOfRecordPrinted`

RecordsRead

OCX:
Determines the number of records actually processed.

`Read& = CrystalReport1.RecordsRead`

RDC:
NumberOfRecordRead
This is a property of the PrintingStatus Object.

`'READ THE RECORDS INTO THE REPORT`
`Report.ReadRecords`

`'PASS THE NUMBER OF RECORDS READ TO THE READ VARIABLE`
`Read& = Report.PrintingStatus.NumberOfRecordRead`

RecordsSelected

OCX:
Determines the number of records selected for inclusion in the report out of the total number of records read.

`Selected& = CrystalReport1.RecordsSelected`

RDC:
NumberOfRecordSelected
This is a property of the PrintingStatus Object.

`'READ THE RECORDS INTO THE REPORT`
`Report.ReadRecords`

`'PASS THE NUMBER OF RECORDS SELECTED TO THE SELECTED VARIABLE`
`Selected& = Report.PrintingStatus.NumberOfRecordSelected`

ReportDisplayPage

OCX:
Indicates which page of a multi-page report is currently being displayed in the preview window.

`Result& = CrystalReport1.ReportDisplayPage`

RDC:
GetCurrentPageNumber
This is a property of the Report Viewer control.
Result% = CRViewer1.GetCurrentPageNumber

ReportFileName

OCX:
Specifies the report to be printed.
CrystalReport1.ReportFileName = "c:\crw\company.rpt"

RDC:
OpenReport
This is a Method of the Application Object and is used for opening reports saved in the Crystal Report format (RPT). The RDC can also open reports saved as ActiveX Designers (.DSR) within Visual Basic.

Opening an RPT.

'General Declarations
'Declare an application object
Dim crxApplication as New craxdrt.Application
'Declare a Report object
Dim Report as craxdrt.Report

'In a function or Sub procedure
Set Report =
  crxApplication.OpenReport("c:\crw\company.rpt", 1)

Declaring a Variable as a New DSR:

'General Declarations
'CrystalReport1 is the name of the DSR in the Designer folder of the Project menu (CrystalReport1.dsr)
Dim Report as New CrystalReport1

Setting a Report Object to a DSR:

'General Declarations
Dim Report as craxdrt.Report

Private Sub Form_Load()
'Set the generic Report object to the DSR
Set Report = New CrystalReport1
End Sub

ReportLatestPage

OCX:
Determines the last page printed in the specified report.
Latest% = CrystalReport1.ReportLatestPage

RDC:
This feature is not implemented in the RDC. An alternative is to create a custom Printer Dialog and store the User’s selection for Start and Last page in variables. For an example of a custom Printer Dialog see the form ‘frmPrintOut’ in the Object Model application at:

- Version 6 of the RDC:
  C:\Program Files\Seagate Crystal Reports\sample\Designer\Object Model App
- Version 7 of the RDC:
  C:\Program Files\SEAGATE SOFTWARE\Crystal Reports\sample\RDC\ObjModel

### ReportSource

**OCX:**
Specifies the source of the report as a report file, a Visual Basic data control, or a True Grid data control.

```vbscript
' Specifies the report source as the TrueDBGrid control
CrystalReport1.ReportSource = 1
```

**RDC:**

Note on Bound Reports:
RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16.

In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply).

### ReportStartPage

**OCX:**
Determines the first page printed in the specified report.

```vbscript
StartPage% = CrystalReport1.ReportStartPage
```

**RDC:**

This feature is not implemented in the RDC. An alternative is to create a custom Printer Dialog and store the User’s selection for Start and Last page in variables. For an example of a custom Printer Dialog see the form ‘frmPrintOut’ in the Object Model application at:

- Version 6 of the RDC:
  C:\Program Files\Seagate Crystal Reports\sample\Designer\Object Model App
- Version 7 of the RDC:
  C:\Program Files\SEAGATE SOFTWARE\Crystal Reports\sample\RDC\ObjModel

### ReportTitle

**OCX:**
Specifies a title for the report.

```vbscript
CrystalReport1.ReportTitle = "My Report"
```

**RDC:**

ReportTitle
This is a property of the Report Object.

```vbscript
Report.ReportTitle = "My Report"
```

### SectionFont

**OCX:**
Sets the font for one or more sections in the specified report.

```vbscript
[form.]Report.SectionFont(SequentialIndex%) [= sectionCode; fontName; size; italic; bold; underline; strikethru$]
```
**RDC:**
The font information for each text or field object can be set through the individual object’s Font object.

<table>
<thead>
<tr>
<th>OCX SectionFont Parameter</th>
<th>RDC Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>sectionCode</td>
<td>Specify the item in the Sections collection of the section to format.</td>
</tr>
<tr>
<td>fontName</td>
<td>Name</td>
</tr>
<tr>
<td>size</td>
<td>Size</td>
</tr>
<tr>
<td>italic</td>
<td>Italic</td>
</tr>
<tr>
<td>bold</td>
<td>Bold</td>
</tr>
<tr>
<td>underline</td>
<td>Underline</td>
</tr>
<tr>
<td>strikethrough</td>
<td>Strikethrough</td>
</tr>
</tbody>
</table>

If the report was saved in a Crystal Report format (RPT) then searching through the section accesses the objects. If the report was saved in an ActiveX Designer format (DSR) the fields can also be formatted in the Format Section event of the DSR.

**RPT**

```vbnet
' Declare a Section object
Dim crxSection As CRAXDRT.Section
' Declare a generic object
Dim crxObject As Object
' Declare a Field object
Dim crxFieldObject As CRAXDRT.FieldObject

' Set the Font properties of the first field in the Report Objects collection of the Detail Section

' Set font name
' Set font size
' Set font italic
' Set font bold
' Set font underline
' Set font strikethrough
```

DSR
Enter the code in the format section event of the desired section. This example sets the fonts of the fields in the Details section.

Private Sub Section3_Format(ByVal pFormattingInfo As Object)
    'Set font name of Field1 through the
    'Font object of crxFieldObject
    Field1.Font.Name = "Arial"
    'Set font size
    Field1.Font.Size = 16
    'Set font italic
    Field1.Font.Italic = True
    'Set font bold
    Field1.Font.Bold = True
    'Set font underline
    Field1.Font.Underline = True
    'set font strikethrough
    Field1.Font.Strikethrough = False

    'Set font name of Field2 through the
    'Font object of crxFieldObject
    Field2.Font.Name = "Arial"
    'Set font size
    Field2.Font.Size = 16
    'Set font italic
    Field2.Font.Italic = True
    'Set font bold
    Field2.Font.Bold = True
    'Set font underline
    Field2.Font.Underline = True
    'set font strikethrough
    Field2.Font.Strikethrough = False

    'Repeat for each field
End Sub

SectionFormat
OCX:
Sets the format for one or more sections in the specified report.
RDC:
The parameters in the OCX SectionFormat property are individual properties of the Section object.

<table>
<thead>
<tr>
<th>OCX SectionFormat Parameter</th>
<th>RDC Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>sectionCode</td>
<td>Specify the item in the Sections collection of the section to format. Report.Sections.Item(&quot;GH&quot;)</td>
</tr>
<tr>
<td>Visible</td>
<td>Suppress</td>
</tr>
<tr>
<td>newPageBefore</td>
<td>NewPageBefore</td>
</tr>
<tr>
<td>newPageAfter</td>
<td>NewPageAfter</td>
</tr>
<tr>
<td>keepTogether</td>
<td>KeepTogether</td>
</tr>
<tr>
<td>suppressBlank</td>
<td>SuppressIfBlank</td>
</tr>
<tr>
<td>resetPageNAfter</td>
<td>ResetPageNumberAfter</td>
</tr>
<tr>
<td>printAtPageBottom</td>
<td>PrintAtBottomOfPage</td>
</tr>
<tr>
<td>underlaySection</td>
<td>UnderlaySection</td>
</tr>
<tr>
<td>backgroundColor</td>
<td>BackColor</td>
</tr>
</tbody>
</table>

If the report was saved in a Crystal Report format (RPT) then the properties set through the Section Object. If the report was saved in an ActiveX Designer format (DSR) the properties can be set through the Section’s property window, in the Section Format Event, or through the Section object.

RPT

```vbnet
'Set the Detail section’s Suppress property
Report.Sections.Item("D").Suppress = False
'Set the section’s NewPageBefore property
Report.Sections.Item("D").NewPageBefore = False
'Set the section’s NewPageAfter property
Report.Sections.Item("D").NewPageAfter = False
'Set the section’s KeepTogether property
Report.Sections.Item("D").KeepTogether = True
'Set the section’s SuppressIfBlank property
Report.Sections.Item("D").SuppressIfBlank = True
'Set the section’s ResetPageNumberAfter property
Report.Sections.Item("D").ResetPageNumberAfter = False
'Set the section’s PrintAtBottomOfPage property
Report.Sections.Item("D").PrintAtBottomOfPage = False
'Set the section’s UnderlaySection property
Report.Sections.Item("D").UnderlaySection = True
'Set the section’s BackColor property
Report.Sections.Item("D").BackColor = vbRed
```
DSR
Enter the code in the format section event of the desired section. This example sets the section properties of the Detail section.

```vbscript
Private Sub Section3_Format(ByVal pFormattingInfo As Object)
    'Set the section’s Suppress property
    Section3.Suppress = False
    'Set the section’s NewPageBefore property
    Section3.NewPageBefore = False
    'Set the section’s NewPageAfter property
    Section3.NewPageAfter = False
    'Set the section’s KeepTogether property
    Section3.KeepTogether = True
    'Set the section’s SuppressIfBlank property
    Section3.SuppressIfBlank = True
    'Set the section’s ResetPageNumberAfter property
    Section3.ResetPageNumberAfter = False
    'Set the section’s PrintAtBottomOfPage property
    Section3.PrintAtBottomOfPage = False
    'Set the section’s UnderlaySection property
    Section3.UnderlaySection = True
    'Set the section’s BackColor property
    Section3.BackColor = vbRed
End Sub
```

**SectionLineHeight**

**OCX:**
Specifies the line height in “twips”. A twip is 1/1440 inch; there are 20 twips in a point.

[form.]Report.SectionLineHeight(SequentialIndex%)[=sectionCode; line; height; ascent$]

```
CrystalReport1.SectionLineHeight(0) = "GH0; 1; 500; 300"
```

**RDC:**
Height
This is a property of the Section Object.
If the report was saved in a Crystal Report format (RPT) then the properties can be set through the Section Object. If the report was saved in an ActiveX Designer format (DSR) the properties can be set through the Section’s property window, in the Section Format Event, or through the Section object.

**RPT**

```
' Set the Detail section’s Height property
Report.Sections.Item("D").Height = 500
```

**DSR**
Enter the code in the format section event of the desired section. This example sets the section Height of the Detail section.

```vba
Private Sub Section3_Format(ByVal pFormattingInfo As Object)
    'Set the section’s Height property
    Section3.Height = 500
End Sub
```

### SectionMinHeight

**OCX:**
Sets the minimum section height for the specified report section.

```vba
CrystalReport1.SectionMinHeight(0) = "DETAIL;500"
```

**RDC:**

MinimumHeight
This is a property of the Section object. It is a read-only property. Use the Height Property to set the section’s height. If the report was saved in a Crystal Report format (RPT) then the properties can be set through the Section Object. If the report was saved in an ActiveX Designer format (DSR) the properties can be set through the Section’s property window, in the Section Format Event, or through the Section object.

**RPT**

```vba
'Set the Detail section’s Height property
Report.Sections.Item("D").Height = 500
```

**DSR**

Enter the code in the format section event of the desired section. This example sets the section Height of the Detail section.

```vba
Private Sub Section3_Format(ByVal pFormattingInfo As Object)
    'Set the section’s Height property
    Section3.Height = 500
End Sub
```

### SelectionFormula

**OCX:**
Specifies the records to be used when printing the report.

```vba
CrystalReport1.SelectionFormula = "{file.QTY} > 5"
```

**RDC:**

RecordSelectionFormula
This is a property of the Report object.

```vba
Report.RecordSelectionFormula = "{file.QTY} > 5"
```

### SessionHandle

**OCX:**
Sets the session handle for a user once the UserName and Password properties have opened an Access.mdb file for use by the report.

```vba
CrystalReport1.SessionHandle = CurrentSessionHandle
```
RDC:
This is not implemented in the RDC. Use SetSessionInfo method of the DatabaseTable object.

```vbscript
'Set the Session info of the first DatabaseTable
'Set in the DatabaseTables collection
Report.Database.Tables.Item(1).SetSessionInfo "User ID", "Password"
```

**SortFields**

OCX:
Specifies the field(s) that are to be used to sort your data when the report is printed.

```vbscript
CrystalReport1.SortFields(0) = "{orders.CUSTOMER}" 
```

RDC:
Field, SortDirection
These are properties of the SortField Object. Field sets the field to sort on, SortDirection sets the sort direction.

```vbscript
'Dim a DatabaseFieldDefinition object
Dim crxDatabaseField As craxdrt.DatabaseFieldDefinition

'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. Accessing the first table to get the 8th field
Set crxDatabaseField = Report.Database.Tables.Item(1).Fields.Item(8)

'Set the field to crxDatabaseField
Report.RecordSortFields.Item(1).Field = crxDatabaseField

'Set the SortField direction
Report.RecordSortFields.Item(1).SortDirection = crAscendingOrder 
```

**SQLQuery**

OCX:
Sets the SQL query string used by the specified report.

```vbscript
CrystalReport1.SQLQuery = "SELECT authors.au_id,
authors.au_lname, authors.au_fname FROM pubs2.dbo.authors
authors WHERE authors.au_lname > 'Madison'"
```

RDC:
SQLQueryString
This is a property of the Report object. It is only available in version 7 of the RDC.

```vbscript
Report.SQLQueryString = "SELECT authors.au_id,
authors.au_lname, authors.au_fname FROM pubs2.dbo.authors
authors WHERE authors.au_lname > 'Madison'"
```
**Status**

**OCX:**
Determines the print status for the specified report.

\[ \text{Status}\% = \text{CrystalReport1.Status} \]

**RDC:**
Progress
This is a property of the PrintingStatus object.

\[ \text{'Print the Report} \]
\[ \text{Report.Printout} \]

\[ \text{'Pass the Progress of the Print job to the Status variable} \]
\[ \text{Status \& = Report.PrintingStatus.Progress} \]

**StoredProcParam**

**OCX:**
Sets the stored procedure parameters when using a report based on SQL stored procedures.

\[ \text{CrystalReport1.StoredProcParam(0)="06/14/1989"} \]

**RDC:**
AddCurrentValue
This is a property of the ParameterFieldDefinition object. All parameters including Stored Procedures and Crystal Parameters are set through the ParameterFieldDefinition object.

\[ \text{'Set the value of the Stored procedure which is the} \]
\[ \text{'first parameter in the ParameterFields collection} \]
\[ \text{Report.ParameterFields.Item(1).AddCurrentValue =} \]
\[ \text{"06/14/1989"} \]

There is no method for setting a stored procedure in RDC Version 6. Refer to Knowledge Base article C2001179 for how to use active data to create a report using the Active Data Driver. By using the Active Data Driver, a developer can set a stored procedure parameter value using the methods of the Data Object, ADO or RDO or DAO or CDO.

**SubreportToChange**

**OCX:**
Specifies whether changes to any of several properties (see list in Remarks below) affect the main report (if you pass an empty string [""] or a subreport (if you pass the name of the subreport).

\[ \text{'Changes to any of the properties apply to the main report} \]
\[ \text{CrystalReport1.SubreportToChange = "} \]
\[ \text{'Changes to any of the properties apply to the Subrpt2 subreport} \]
\[ \text{CrystalReport1.SubreportToChange = "Subrpt2"} \]

**RDC:**
OpenSubreport
This is a method of the SubreportObject object and the Report Object.

\[ \text{Report.OpenSubreport} \]
Pass the name of the subreport to the OpenSubreport Method of the Report object. This method is valid for reports saved in the Crystal Report format (RPT) or as an ActiveX Designer (DSR) within Visual Basic.

```vbscript
'Declare Report object to set to the Subreport
Dim crxSubreport As CRAXDRT.Report

'Set crxSubreport to the subreport "sub1"
Set crxSubreport = Report.OpenSubreport("sub1")
```

SubreportObject.OpenSubreport

Open the subreport by setting a subreport to a valid SubreportObject in the report. Searching through the sections, and report objects in the section accesses the subreport. Valid for RPT and DSR.

```vbscript
'Search through each section of the report
For Each crxSection In Report.Sections
    'Search through each report object in each section
    For Each crxObject In crxSection.ReportObjects
        'If the report object is a subreport set it to crxSubreportObject
        If crxObject.Kind = crxSubreportObject Then
            Set crxSubreportObject = crxObject
            'Set crxSubreport to the subreport
            Set crxSubreport = crxSubreportObject.OpenSubreport
            'Enter code for subreport
            End If
        Next crxObject
    Next crxSection

'If the location of the subreport is known, the amount of code can be reduced. For example, the subreport is the first report object in the first section (Report Header) of the report. Valid for RPT and DSR.

'Search through each section of the report
For Each crxSection In Report.Sections
    'Search through each report object in each section
    For Each crxObject In crxSection.ReportObjects
        'If the report object is a subreport set it to crxSubreportObject
        If crxObject.Kind = crxSubreportObject Then
            Set crxSubreportObject = crxObject
            'Set crxSubreport to the subreport
            Set crxSubreport = crxSubreportObject.OpenSubreport
            'Enter code for subreport
            End If
        Next crxObject
    Next crxSection
```

'Set crxSubreport to the first object in the Report Header section.'


If a report is instantiated by declaring a variable as a new DSR the subreport object can be accessed directly without having to search through the sections.

'General Declarations
'Declare a SubreportObject Object

'Sub or function

'Declare Report object to set to the Subreport
Dim crxSubreport As CRAXDRT.Report

'Set crxSubreport to the subreport
Set crxSubreport = Report.Subreport1.OpenSubreport

**UserName**

**OCX:**
Enters the name given to a user for logging on to a protected Access.mdb file to obtain data files needed by the report.

```vbnet
CrystalReport1.UserName = "MIS"
```

**RDC:**
SetSessionInfo
This is the first parameter in the SetSessionInfo method of the DatabaseTable object.

```vbnet
'Set the session info for the first table in the DatabaseTables collection
Report.Database.Tables.Item(1).SetSessionInfo "User ID", "Password"
```

**WindowAllowDrillDown**

**OCX:**
Indicates whether or not drill-down on summary values is allowed in the preview window.

```vbnet
CrystalReport1.WindowAllowDrillDown = FALSE
```

**RDC:**
EnableDrillDown
This is a property of the Report Viewer control.

```vbnet
CRViewer1.EnableDrillDown = True
```

**WindowBorderStyle**

**OCX:**
Specifies the type of border for the preview window.

```vbnet
CrystalReport1.WindowBorderStyle = 2
```
RDC:
The Report Viewer control is embedded on a form. Set the Border Style of the Report Viewer control parent form.

Form1.BorderStyle = 2

WindowControlBox
OCX:
Specifies whether or not the preview window is to have a control (system menu) box in the upper left-hand corner when the report is printed to a window.

CrystalReport1.WindowControlBox = True

RDC:
The Report Viewer control is embedded on a form. Set the Border Style of the Report Viewer control parent form’s Property window.

WindowControls
OCX:
Specifies whether or not the print controls are to appear in the preview window when printing a report to a window.

CrystalReport1.WindowControls = True

RDC:
EnableToolbar
This is a property of the Report Viewer control.

CRViewer1.EnableToolbar = True

WindowHeight
OCX:
Sets the height of the preview window when the report is printed to a window.

CrystalReport1.WindowHeight = 300

RDC:
Height
This is a property of the Report Viewer control and sets the height of the control within the parent form.

'To set the height to match the parent form
'Place within the Form Resize event.

CRViewer1.Height = ScaleHeight

WindowLeft
OCX:
Sets the distance, in pixels, that the preview window is to appear from the left edge of the parent window. If the preview window is a top-level window, then the distance is measured from the left edge of the screen. (A standard VGA monitor is 640 x 480 pixels).

CrystalReport1.WindowLeft = 100

RDC:
Left
This is a property of the Report Viewer control and sets the left side of the control within the parent form.

'To set the control to the left edge of the parent form
'Place within the Form Resize event.

CRViewer1.Left = 0
**WindowMaxButton**

**OCX:**
Specifies whether or not the preview window is to have an active, hidden or grayed-out maximize button when the report is printed to a window.

```vbnet
CrystalReport1.WindowMaxButton = False
```

**RDC:**
The Report Viewer control is embedded on a form. Set the MaxButton of the Report Viewer control parent form’s property window.

**WindowMinButton**

**OCX:**
Specifies whether or not the preview window is to have an active, hidden or grayed-out minimize button when the report is printed to a window.

```vbnet
CrystalReport1.WindowMinButton = True
```

**RDC:**
The Report Viewer control is embedded on a form. Set the MinButton of the Report Viewer control parent form’s property window.

**WindowParentHandle**

**OCX:**
Specifies the handle of the parent window if the preview window is to be the child of another window.

```vbnet
CrystalReport1.WindowParentHandle = Form1.hWnd
```

**RDC:**
The Report Viewer control is embedded on a form. The form can be made a child of an MDI form.

**WindowShowCancelBtn**

**OCX:**
Indicates whether or not a Cancel button is available in the preview window.

```vbnet
CrystalReport1.WindowShowCancelBtn = False
```

**RDC:**
EnableStopButton
This is a property of the Report Viewer control.

```vbnet
CRViewer1.EnableStopButton = True
```

**WindowShowCloseBtn**

**OCX:**
Indicates whether or not a Close button is available in the preview window.

```vbnet
CrystalReport1.WindowShowCloseBtn = True
```

**RDC:**
EnableCloseButton
This is a property of the Report Viewer control. The Close Button on the Report Viewer control only closes the report views within report.

```vbnet
CRViewer1.EnableCloseButton = True
```

**WindowShowExportBtn**
OCX:  
Indicates whether or not an Export button is available in the preview window.  
\[ \text{CrystalReport1.WindowShowExportBtn} = \text{True} \]

RDC: 
EnableExportButton  
This is a property of the Report Viewer control.  
\[ \text{CRViewer1.EnableExportButton} = \text{True} \]

WindowShowGroupTree

OCX:  
Indicates whether or not a Group Tree is displayed in the preview window.  
\[ \text{CrystalReport1.WindowShowGroupTree} = \text{False} \]

RDC: 
EnableGroupTree  
This is a property of the Report Viewer control.  
\[ \text{CRViewer1.EnableGroupTree} = \text{True} \]

WindowShowNavigationCtls

OCX:  
Indicates whether or not the Navigation Controls are available in the preview window.  
\[ \text{CrystalReport1.WindowShowNavigationCtls} = \text{True} \]

RDC: 
EnableNavigationControls  
This is a property of the Report Viewer control.  
\[ \text{CRViewer1.EnableNavigationControls} = \text{True} \]

WindowShowPrintBtn

OCX:  
Indicates whether or not a Print button is available in the preview window.  
\[ \text{CrystalReport1.WindowShowPrintBtn} = \text{False} \]

RDC: 
EnablePrintButton  
This is a property of the Report Viewer control.  
\[ \text{CRViewer1.EnablePrintButton} = \text{True} \]

WindowShowPrintSetupBtn

OCX:  
Indicates whether or not a Print Setup button is available in the preview window.  
\[ \text{CrystalReport1.WindowShowPrintSetupBtn} = \text{False} \]

RDC:  
This feature is not implemented in the Report Viewer control for RDC versions 6 and 7. You can call the Microsoft Common Dialog from the PrintButtonClicked event or use your own Custom Dialog. For an example of a custom Printer Dialog see the form 'frmPrintOut' in the Object Model application at:  
- RDC version 6  
  - C:\Program Files\Seagate Crystal Reports\sample\Designer\Object Model App
• RDC version 7
  - C:\Program Files\SEAGATE SOFTWARE\Crystal Reports\sample\RDC\ObjModel

Using the Microsoft CommonDialog

'Add a CommonDialog control to the form containing the Crystal Report Report Viewer control.

Private Sub CRViewer1_PrintButtonClicked(UseDefault As Boolean)
  'Set default to false so report 'does not print twice
  UseDefault = False

  'display the printer dialog
  CommonDialog1.ShowPrinter

  'Print the report without prompting user
  'All changes made in the printer dialog 'will be reflected in the report
  Report.PrintOut False
End Sub

• Version 8 of the RDC

'PrinterSetup provides a Windows standard printer setup window to allow the user 'to change the printer properties directly at runtime.

'Call the Printer Setup dialog box
Report.PrinterSetup Me.hWnd

**WindowShowProgressCtls**

OCX:
Determines whether or not controls indicating the progress of a report being generated are displayed in the preview window.

CrystalReport1.WindowShowProgressCtls = False

RDC:
EnableProgressControl
This is a property of the Report Viewer control.

CRViewer1.EnableProgressControl = True

**WindowShowRefreshBtn**

OCX:
Indicates whether or not a Refresh button is available in the preview window.

CrystalReport1.WindowShowRefreshBtn = False
RDC:
EnableRefreshButton
This is a property of the Report Viewer control.

\[
\text{CRViewer1.EnableRefreshButton = True}
\]

**WindowShowSearchBtn**

**OCX:**
Indicates whether or not a Search button is available in the preview window.

\[
\text{CrystalReport1.WindowShowSearchBtn = True}
\]

RDC:
EnableSearchControl
This is a property of the Report Viewer control.

\[
\text{CRViewer1.EnableSearchControl = True}
\]

**WindowShowZoomCtl**

**OCX:**
Indicates whether or not Zoom controls are available in the preview window.

\[
\text{CrystalReport1.WindowShowZoomCtl = True}
\]

RDC:
EnableZoomControl
This is a property of the Report Viewer control.

\[
\text{CRViewer1.EnableZoomControl = True}
\]

**WindowState**

**OCX:**
Sets the state of the preview window as normal, minimized or maximized when the report is printed.

\[
\text{CrystalReport1.WindowState = 2}
\]

RDC:
The Report Viewer control is embedded on a form. Sets the State of the Report Viewer control parent form.

\[
\text{Form1.WindowState = 2}
\]

**WindowTitle**

**OCX:**
Specifies the title to appear in the preview window title bar when the report is printed to a window.

\[
\text{CrystalReport1.WindowTitle = "Quarterly Earnings"}
\]

RDC:
The Report Viewer control is embedded on a form. Sets the Caption of the Report Viewer control parent form.

\[
\text{Form1.Caption = "Quarterly Earnings"}
\]

**WindowTop**

**OCX:**
Sets the distance, in pixels, that the preview window is to appear from the top edge of the parent window. If the preview window is a top-level window, then the distance is measured from the top edge of the screen.

\[
\text{CrystalReport1.WindowLeft = 100}
\]
RDC:
Top
This is a property of the Report Viewer control and sets the topside of the control within the parent form.

  'To set the control to the left edge of the parent form
  'Place within the Form Resize event.
  CRViewer1.Top = 0

WindowWidth

OCX:
Specifies the width of the preview window in pixels.

  CrystalReport1.WindowWidth = 480

RDC:
Width
This is a property of the Report Viewer control and sets the width of the control within the parent form.

  'To set the width to match the parent form
  'Place within the Form Resize event.
  CRViewer1.Width = ScaleHeight

Methods

FetchSelectionFormula

OCX:
FetchSelectionFormula returns the selection formula from the current report.

  SelectionFormula$ = CrystalReport1.FetchSelectionFormula

RDC:
RecordSelectionFormula
This is a read and write property of the Report object.

  SelectionFormula$ = Report.RecordSelectionFormula

GetNSubreports

OCX:
Looks at the report specified in ReportFileName, and returns the number of subreports in that report.

  Number = CrystalReport1.GetNSubreports

RDC:
To get the number of subreports, search for each subreport object in each section of the Report. See the OCX property SubreportToChange for the RDC methods of opening a subreport.

  'Declare a Section object
  Dim crxSection As CRAXDRT.Section
  'Declare a Generic object
  Dim crxObject As Object
  'Declare a variable to hold the number of subreports
  Dim nSubreports as Integer
'Search through each section of the report
For Each crxSection In Report.Sections
    'Search through each report object in each section
    For Each crxObject In crxSection.ReportObjects
        'If the report object is a subreport
        'Add 1 to nSubreports
        If crxObject.Kind = crSubreportObject Then
            nSubreports = nSubreports + 1
        End If
    Next crxObject
Next crxSection

GetNthSubreportName

OCX:
Looks at the report specified in the ReportFileName property and returns a string which is the name of the nth subreport in that report.

    SubreportName=CrystalReport1.GetNthSubreportName (2)

RDC:
SubreportName
This is a property of the SubreportObject.

    'Declare a variable to hold the name of the subreport
    Dim strSubreport as String

    'The subreport is the first Report object in the Report Footer
    'section
    strSubreport = Report.Sections.Item("RF").ReportObjects.Item(1).SubreportName

LogonServer

OCX:
Logs on to the specified server and returns a unique connection ID which can be used to log off of this server using the LogoffServer.

    connectionId% = CrystalReport1.LogonServer ("pdsodbc.dll", "Accounting", "Administration", "bobg", "bigboard")

RDC:
LogonServer
This is a method of the Application and Database Objects.

    CrxApplication.LogonServer "pdsodbc.dll", "Accounting", "Administration", "bobg", "bigboard"
    'Log onto the data source
**PageCount**

**OCX:**
Returns the number of pages in the report.

\[
\text{NumberOfPages} = \text{CrystalReport1.PageCount}
\]

**RDC:**
NumberOfPages
This is a property of the PrintingStatus Object.

\`
'Read the records into the report
Report.ReadRecords

'Pass the number of records read to the Read variable
Pages& = Report.PrintingStatus.NumberOfPages
\`

**PageFirst**

**OCX:**
Displays the first page of the report in the preview window.

\[
\text{CrystalReport1.PageFirst}
\]

**RDC:**
ShowFirstPage
This is a method of the Report Viewer control.

\[
\text{CRViewer1.ShowFirstPage}
\]

**PageLast**

**OCX:**
Displays the last page of the report in the preview window.

\[
\text{CrystalReport1.PageLast}
\]

**RDC:**
ShowLastPage
This is a method of the Report Viewer control.

\[
\text{CRViewer1.ShowLastPage}
\]

**PageNext**

**OCX:**
Displays the next page of the report in the preview window.

\[
\text{CrystalReport1.PageNext}
\]

**RDC:**
ShowNextPage
This is a method of the Report Viewer control.

\[
\text{CRViewer1.ShowNextPage}
\]

**PagePrevious**
OCX:
Displays the previous page of the report in the preview window.
\[\text{CrystalReport1.PagePrevious}\]

RDC:
ShowPreviousPage
This is a method of the Report Viewer control.
\[\text{CRViewer1.ShowPreviousPage}\]

PageShow
OCX:
Displays a specific page of the report in the preview window.
\[\text{CrystalReport1.PageShow (3)}\]

RDC:
ShowNthPage
This is a method of the Report Viewer control.
\[\text{CRViewer1.ShowNthPage 3}\]

If the report is to go to a specific page when first previewed, the control must be allowed to complete the downloading of data before setting the page.

'Set the Report to the Report Viewer
\[\text{CRViewer1.ReportSource} = \text{Report}\]

'Show the Report
\[\text{CRViewer1.ViewReport}\]

'SContinue loop while the Report Viewer is busy downloading data
\[\text{While CRViewer1.IsBusy}\]
\[\text{DoEvents}\]
\[\text{Wend}\]

'Set the report to preview on the third page
\[\text{CRViewer1.ShowNthPage 3}\]

PageZoom
OCX:
Sets the magnification factor for the report in the preview window to a value between 25% and 400% of the actual size.
\[\text{CrystalReport1.PageZoom (150)}\]

RDC:
Zoom
This is a method of the Report Viewer control.
\[\text{CRViewer1.Zoom 150}\]
If the report is set to a specific zoom level when it is first previewed, the downloading of data must be completed before setting the page.

```
'Set the Report to the Report Viewer
CRViewer1.ReportSource = Report

'View the Report
CRViewer1.ViewReport

'Continue loop while the Report Viewer is busy downloading data
While CRViewer1.IsBusy
    DoEvents
Wend

'Set the report’s zoom level to 150%
CRViewer1.Zoom 150
```

**PageZoomNext**

**OCX:**
Sets the magnification level of the report in a preview window to the next default zoom level.

```
CrystalReport1.PageZoomNext
```

**RDC:**
This method is not available in the Report Viewer control, however the functionality can be duplicated in code.

A button is placed on a form containing the Report Viewer. The Report opens at the Whole Page zoom level. Clicking the button will cycle through the zoom levels.

```
Private Sub Command1_Click()
    'ZoomLvl is initialized to 0 in the Form_Load

    'ZoomArray is a ten element array initialized in the Form_Load
    'and containing all the zoom levels available from the 'Zoom Level Drop Down List on the Report Viewer form.
    'Whole Page’ = 1, ‘Page Width’ = 2

    'When ZoomLvl reaches the top Level
    'reset to the bottom level
    If ZoomLvl < 9 Then
        ZoomLvl = ZoomLvl + 1
    Else
        ZoomLvl = 0
    End If
```
'Set Zoom to the Next Level
CRViewer1.Zoom (ZoomArray(ZoomLvl))

End Sub

**PrinterSelect**

**OCX:**
Displays the Printer Selection common dialog box, which enables the user to specify a different printer.

CrystalReport1.PrinterSelect

**RDC:**
To display a Printer Selection common dialog box:

'Add a Microsoft Common Dialog Control to the form containing the Report Viewer
'Add the following code to the PrintButton Clicked event of the Report Viewer
Private Sub CRViewer1_PrintButtonClicked(UseDefault As Boolean)
    CommonDialog1.ShowPrinter
End Sub

**PrintReport**

**OCX:**
PrintReport triggers the printing of the report.

Result% = CrystalReport1.PrintReport

**RDC:**
The RDC prints to screen, printer and export through three different methods, all of which are methods of the Report object.

Print to screen:
Viewing the report is done through the Report Viewer control:

CRViewer1.ReportSource = Report
CRViewer1.ViewReport

Printer:
Print the report without prompting the user.
Report.Printout False

Export:
Export the report without prompting the user.
Report.Export False

**ReplaceSelectionFormula**

**OCX:**
ReplaceSelectionFormula overrides the selection formula from the current report with the string that is passed.

CrystalReport1.ReplaceSelectionFormula
("{Company.State}='CA'")

**RDC:**
RecordSelectionFormula
This is a property of the Report object.

\[
\text{Report.RecordSelectionFormula} = \{\text{Company.State}\} = \text{CA}'
\]

Reset

**OCX:**
Resets the value of all properties (except DataSource Property) to their default values.

\[
\text{CrystalReport1.Reset}
\]

**RDC:**
To Reset the report, set the Report object to “Nothing”, then open the report again and set the new properties. See the OCX property ReportFileName for the RDC methods for opening reports.

\[
\text{Set the Report object to Nothing}
\]

\[
\text{Report} = \text{Nothing}
\]

RetrieveDataFiles

**OCX:**
RetrieveDatafiles retrieves all "table" locations from the current report, populates Datafiles and returns the number of "tables" in the report.

\[
\text{NumberOfDatafiles} = \text{CrystalReport1.RetrieveDatafiles}
\]

**RDC:**
Location
This is a method of the DatabaseTable object.

\[
\text{Get the number of tables in the report}
\]

\[
\text{from the count property of the DatabaseTables collection}
\]

\[
\text{nTables} = \text{Report.Database.Tables.Count}
\]

\[
\text{Set the location the first DatabaseTable in the DatabaseTables collection}
\]

\[
\text{Report.Database.Tables.Item(1).DatabaseTable.Location} = \text{"c:\new\xtreme.mdb"}
\]

RetrieveLogonInfo

**OCX:**
RetrieveLogonInfo retrieves logon information (except for the password) for all "tables" in the current report, populates LogOnInfo Property and returns the number of "tables" in the report.

\[
\text{NumberOfTables} = \text{CrystalReport1.RetrieveLogonInfo}
\]

**RDC:**
Use the SetLogonInfo to connect to the server. To read the connection information, use the following properties from the DatabaseTable object.

\[
\text{Get the number of tables in the report}
\]

\[
\text{from the count property of the DatabaseTables collection}
\]

\[
\text{nTables} = \text{Report.Database.Tables.Count}
\]

\[
\text{Set the location the first DatabaseTable in the DatabaseTables collection}
\]

**RetrieveSQLQuery**

**OCX:**
RetrieveSQLQuery retrieves the SQL Query from the current report and populates SQLQuery Property.

`CrystalReport1.RetrieveSQLQuery`

**RDC:**
SQLQueryString
This is a property of the Report Object.

```vbnet
' Declare a string variable to hold the SQL Query
Dim strSQL As String

' To retrieve the SQL Query
strSQL = Report.SQLQueryString

' To pass a SQL Query
Report.SQLQueryString = strSQL
```

**RetrieveStoredProcParams**

**OCX:**
RetrieveStoredProcParams retrieves all stored procedure parameters from the current report, populates StoredProcParam Property and returns the number of parameters.

`NumberofParams% = CrystalReport1.RetrieveStoredProcParams`

**RDC:**
All Stored Procedure parameters are part of the ParameterFieldDefinition collection. Use the following properties of the ParameterFieldDefinition object and ParameterFieldDefinitions collection.

```vbnet
' Retrieve the first default value in the parameter
' RDC has the ability to add multiple values
nParameterValue = Report.Parameters.Item(1).GetNthDefaultValue(1)

' Set the first parameter in the ParameterFields collection
Report.Parameters.Item(1).AddCurrentValue nParameterValue

' Get the number of parameters in the report
' from the count property of the ParameterFields collection
nParameters% = Report.ParameterFields.Count
```

**SetTablePrivateData**

**OCX:**
SetTablePrivateData sets the report data to the recordset in memory. Use with Crystal Data Object, ActiveX Data Object 1.5 and higher, Remote Data Object 2.0, etc.
CrystalReport1.SetTablePrivateData 0, 3, ADOrs

**RDC:**
SetPrivateData
SetPrivateData is a property of the DatabaseTable object.

`'Pass the recordset to the first table in the`  
`'DatabaseTables collection`  
`'3 is the Data Tag and is currently the`  
`'only valid value. rs is any valid recordset`  
Report.Database.Tables.Item(1).SetPrivateData 3, rs

**SpecifyDataSourceField**

**OCX:**
Enables you to specify the columns that appear, their order and their width for reports that are automatically generated from a Data control. If you call this function one or more times, only the columns indicated by the calls will appear in the report. You must call this function one time for each column you are setting.

CrystalReport1.SpecifyDataSourceField 0,"Year Born",10

**RDC:**

**Note on Bound Reports:**
RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16.

In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply).
## Appendix D: Features Comparison of OCX to RDC

<table>
<thead>
<tr>
<th>OCX Methods and Properties</th>
<th>RDC Method</th>
<th>RDC Object</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Properties:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>N/A</td>
<td>N/A</td>
<td>To Print use the PrintOut method of the Report object. To Preview use the ViewReport method of the CRViewer. To Export use the Export method of the Report object.</td>
</tr>
<tr>
<td>BoundReportFooter</td>
<td>Suppress</td>
<td>Section</td>
<td><strong>Note on Bound Reports:</strong> RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16. In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply). For more information, please go to page 17.</td>
</tr>
<tr>
<td>BoundReportHeading</td>
<td>ReportTitle</td>
<td>Report</td>
<td>See above note on Bound Reports</td>
</tr>
<tr>
<td>Connect</td>
<td>SetLogonInfo</td>
<td>DatabaseTable</td>
<td></td>
</tr>
<tr>
<td>CopiesToPrinter</td>
<td>Printout</td>
<td>Report</td>
<td>An optional second parameter of the Printout method.</td>
</tr>
<tr>
<td>DataFiles</td>
<td>Location</td>
<td>DatabaseTable</td>
<td></td>
</tr>
<tr>
<td>DataSource</td>
<td>N/A</td>
<td>N/A</td>
<td>See above note on Bound Reports</td>
</tr>
<tr>
<td>Destination</td>
<td>N/A</td>
<td>N/A</td>
<td>There are separate commands to print Preview or Export.</td>
</tr>
<tr>
<td>DetailCopies</td>
<td>CopiesToPrint</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>DialogParentHandle</td>
<td>SetDialogParentWindow</td>
<td>Report</td>
<td></td>
</tr>
<tr>
<td>DiscardSavedData</td>
<td>DiscardSavedData</td>
<td>Report</td>
<td></td>
</tr>
<tr>
<td>EmailCCList</td>
<td>MailCcList</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>EmailMessage</td>
<td>MailMessage</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>EmailSubject</td>
<td>MailSubject</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>EmailToList</td>
<td>MailToList</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>EmailVIMBCCList</td>
<td>MailBccList</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>ExchangeFolder</td>
<td>ExchangeFolderPath</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>ExchangeProfile</td>
<td>ExchangeProfile</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>ExchangePassword</td>
<td>ExchangePassword</td>
<td>ExportOptions</td>
<td></td>
</tr>
<tr>
<td>Formulas</td>
<td>Text</td>
<td>FormulaFieldDefinition</td>
<td>Access the individual Formula from the FormulaFieldDefinitions collection.</td>
</tr>
<tr>
<td>GraphData</td>
<td>N/A</td>
<td>N/A</td>
<td>The GraphData property is not included in the OCX as of version 7, and is not included in the RDC.</td>
</tr>
<tr>
<td>GraphOptions</td>
<td>See notes</td>
<td>GraphObject</td>
<td>The parameters of the GraphOptions property are individual properties of the GraphObject. See below for the applicable GraphOptions parameter to GraphObject property match. FontFace - N/A BarDirection - GraphDirection LabelRisers – DataPoint</td>
</tr>
</tbody>
</table>
### GridLines – GroupAxisGridline
- **Legend** – EnableShowLegend
- **Max** – MaxDataAxisValue
- **Min** – MinDataAxisValue

### GraphText
- **See notes**
- **GraphObject**
  - The parameters of the GraphText property are individual properties of the GraphObject. See below for the applicable GraphText parameter to GraphObject property match.
  - **Title** – Title
  - **SubTitle** – SubTitle
  - **FootNote** – FootNote
  - **Series** – SeriesTitle
  - **Group** – GroupsTitle
  - **X** – XaxisTitle
  - **Y** – YaxisTitle
  - **Z** – ZaxisTitle

### GroupCondition
- **See notes**
- **Area**
  - The parameters of the GroupCondition property are individual properties of the Area object. See below for the applicable GroupCondition parameter to Area property match.
  - **Field** – GroupConditionField
  - **Condition** – GroupCondition
  - **SortDirection** – SortDirection

### GroupSelectionFormula
- **Add**
- **GroupSortFields**
  - Adds the SummaryField to the GroupSortFields collection.

### LastErrorNumber
- **N/A**
- **N/A**
  - Use Visual Basic Err object. Err.Number

### LastErrorString
- **N/A**
- **N/A**
  - Use Visual Basic Err object. Err.Description

### LogOnInfo
- **SetLogOnInfo**
- **DatabaseTable**

### MarginBottom
- **BottomMargin**
- **Report**

### MarginLeft
- **LeftMargin**
- **Report**

### MarginRight
- **RightMargin**
- **Report**

### MarginTop
- **TopMargin**
- **Report**

### ParameterFields
- **AddCurrentValue**
- **ParameterFieldDefinition**
  - Access the individual parameter through the ParameterFieldDefinitions collection. The RDC takes advantage of the enhanced parameter fields of Seagate Crystal Reports 7 and is able to pass multiple values as well.

### Password
- **SetSessionInfo**
- **DatabaseTable**
  - Second parameter of the SetSessionInfo method.

### PrintDay
- **PrintDate**
- **Report**
  - Sets the entire print date.

### PrinterCollation
- **PrintOut**
- **Report**
  - Optional third parameter of the PrintOut method.

### PrinterCopies
- **PrintOut**
- **Report**
  - Optional second parameter of PrintOut method.

### PrinterDriver
- **SelectPrinter**
- **Report**
  - First parameter of SelectPrinter method.

### PrinterName
- **SelectPrinter**
- **Report**
  - Second parameter of SelectPrinter method.

### PrinterPort
- **SelectPrinter**
- **Report**
  - Third parameter of SelectPrinter method.

### PrinterStartPage
- **PrintOut**
- **Report**
  - Optional fourth parameter of the PrintOut method.

### PrinterStopPage
- **PrintOut**
- **Report**
  - Optional fifth parameter of the PrintOut method.

### PrintFileCharSepQuote
- **CharStringDelimiter**
- **ExportOptions**

### PrintFileCharSepSeparator
- **CharFieldDelimiter**
- **ExportOptions**
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrintFileLinesPerPage</td>
<td>Number of lines per page</td>
<td></td>
</tr>
<tr>
<td>PrintFileName</td>
<td>Disk file name</td>
<td></td>
</tr>
<tr>
<td>PrintFileODBCPassword</td>
<td>ODBC data source password</td>
<td></td>
</tr>
<tr>
<td>PrintFileODBCSource</td>
<td>ODBC data source name</td>
<td></td>
</tr>
<tr>
<td>PrintFileODBCTable</td>
<td>ODBC export table name</td>
<td></td>
</tr>
<tr>
<td>PrintFileODBCUser</td>
<td>ODBC data source user ID</td>
<td></td>
</tr>
<tr>
<td>PrintFileType</td>
<td>Format type</td>
<td></td>
</tr>
<tr>
<td>PrintFileUseRptDateFmt</td>
<td>Use report date format</td>
<td>Sets the entire print date.</td>
</tr>
<tr>
<td>PrintFileUseRptNumberFmt</td>
<td>Use report number format</td>
<td>Sets the entire print date.</td>
</tr>
<tr>
<td>PrintMonth</td>
<td>Print date report</td>
<td>Sets the entire print date.</td>
</tr>
<tr>
<td>ProgressDialog</td>
<td>Display progress dialog</td>
<td></td>
</tr>
<tr>
<td>RecordsPrinted</td>
<td>Number of records printed</td>
<td></td>
</tr>
<tr>
<td>RecordsRead</td>
<td>Number of record</td>
<td></td>
</tr>
<tr>
<td>RecordsSelected</td>
<td>Number of records selected</td>
<td></td>
</tr>
<tr>
<td>ReportDisplayPage</td>
<td>Get page number</td>
<td>CRViewer</td>
</tr>
<tr>
<td>ReportFileName</td>
<td>Open report report</td>
<td></td>
</tr>
<tr>
<td>ReportLatestPage</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ReportSource</td>
<td>N/A</td>
<td>Note on Bound Reports: RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16. In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply) For more information, please go to page 17.</td>
</tr>
<tr>
<td>ReportStartPage</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

This feature is not implemented in the RDC. An alternative is to create a custom Printer Dialog and store the User’s selection for Start and Last page in variables. For an example of a custom Printer Dialog see the form ‘frmPrintOut’ in the Object Model application at:

- Version 6 of the RDC: C:\Program Files\Seagate Crystal Reports\sample\Designer\Object Model App
- Version 7 and 8 of the RDC: C:\Program Files\SEAGATE SOFTWARE\Crystal Reports\sample\RDC\ObjModel

Note on Bound Reports: RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16. In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply) For more information, please go to page 17.
<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportTitle</td>
<td>ReportTitle</td>
<td>Report</td>
</tr>
<tr>
<td>SectionFont</td>
<td>See notes</td>
<td>See Notes</td>
</tr>
<tr>
<td>SectionFormat</td>
<td>See notes</td>
<td>Section</td>
</tr>
<tr>
<td>SectionLineHeight</td>
<td>See Section Format notes</td>
<td>Section</td>
</tr>
<tr>
<td>SectionMinHeight</td>
<td>See Section Format notes</td>
<td>Section</td>
</tr>
<tr>
<td>SelectionFormula</td>
<td>RecordSelectionFormula</td>
<td>Report</td>
</tr>
<tr>
<td>SessionHandle</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SQLQuery</td>
<td>SQLQueryString</td>
<td>Report</td>
</tr>
<tr>
<td>Status</td>
<td>Progress</td>
<td>PrintingStatus</td>
</tr>
<tr>
<td>StoredProcParam</td>
<td>AddCurrentValue</td>
<td>ParameterFieldDefinition</td>
</tr>
<tr>
<td>SubreportToChange</td>
<td>OpenSubreport</td>
<td>Report and SubreportObject</td>
</tr>
<tr>
<td>UserName</td>
<td>SetSessionInfo</td>
<td>DatabaseTable</td>
</tr>
<tr>
<td>WindowAllowDrillDown</td>
<td>EnableDrillDown</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowBorderStyle</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowControlBox</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowControls</td>
<td>EnableToolbar</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowHeight</td>
<td>Height</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowLeft</td>
<td>Left</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowMaxButton</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowMinButton</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowParentHandle</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowShowCancelBtn</td>
<td>EnableStopButton</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowCloseBtn</td>
<td>EnableCloseButton</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowExportBtn</td>
<td>EnableExportButton</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowGroupTree</td>
<td>EnableGroupTree</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowNavigationCtls</td>
<td>EnableNavigationControls</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowPrintBtn</td>
<td>EnablePrintButton</td>
<td>CRViewer</td>
</tr>
<tr>
<td>WindowShowPrintSetupBtn</td>
<td>PrinterSetup (RDC version 8 only).</td>
<td>Report</td>
</tr>
</tbody>
</table>
|                          |                        | This feature is not implemented in the CRViewer for RDC versions 6 and 7. You can call the Microsoft Common Dialog from the PrintButtonClicked event or use your...
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WindowShowProgressCtls</td>
<td>EnableProgressControl</td>
</tr>
<tr>
<td>WindowShowRefreshBtn</td>
<td>EnableRefreshButton</td>
</tr>
<tr>
<td>WindowShowSearchBtn</td>
<td>EnableSearchControl</td>
</tr>
<tr>
<td>WindowShowZoomCtl</td>
<td>EnableZoomControl</td>
</tr>
<tr>
<td>WindowState</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowTitle</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowTop</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowWidth</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowState</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowTitle</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowTop</td>
<td>N/A</td>
</tr>
<tr>
<td>WindowWidth</td>
<td>N/A</td>
</tr>
<tr>
<td>Methods:</td>
<td></td>
</tr>
<tr>
<td>FetchSelectionFormula</td>
<td>RecordSelectionFormula</td>
</tr>
<tr>
<td>GetNSubreports</td>
<td>N/A</td>
</tr>
<tr>
<td>GetNthSubreportName</td>
<td>Name</td>
</tr>
<tr>
<td>LogonServer</td>
<td>LogonServer</td>
</tr>
<tr>
<td>PageCount</td>
<td>NumberOfPages</td>
</tr>
<tr>
<td>PageFirst</td>
<td>ShowFirstPage</td>
</tr>
<tr>
<td>PageLast</td>
<td>ShowLastPage</td>
</tr>
<tr>
<td>PageNext</td>
<td>ShowNextPage</td>
</tr>
<tr>
<td>PagePrevious</td>
<td>ShowPreviousPage</td>
</tr>
<tr>
<td>PageShow</td>
<td>ShowNthPage</td>
</tr>
<tr>
<td>PageZoom</td>
<td>Zoom</td>
</tr>
<tr>
<td>PageZoomNext</td>
<td>N/A</td>
</tr>
<tr>
<td>PrinterSelect</td>
<td>N/A</td>
</tr>
<tr>
<td>PrintReport</td>
<td>N/A</td>
</tr>
<tr>
<td>ReplaceSelectionFormula</td>
<td>RecordSelectionFormula</td>
</tr>
<tr>
<td>Reset</td>
<td>N/A</td>
</tr>
<tr>
<td>RetrieveDataFiles</td>
<td>Location</td>
</tr>
<tr>
<td>RetrieveLogonInfo</td>
<td>SetLogonInfo</td>
</tr>
<tr>
<td>RetrieveSQLQuery</td>
<td>SQLQueryString</td>
</tr>
</tbody>
</table>
| Method Name                      | Count/Get
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RetrieveStoredProcParams</td>
<td>Count, Get nth Current Value, Add Current Value</td>
</tr>
<tr>
<td>ParameterFieldDefinitions</td>
<td>ParameterFieldDefinition</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Method Name                      | Count/Get
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTablePrivateData</td>
<td>SetPrivateData and SetDataSource</td>
</tr>
<tr>
<td>DatabaseTable and Database</td>
<td>SetPrivateData is a property of the DatabaseTable object and can be used for each table. SetDataSource is a property of the Database object and is used to pass a single Recordset.</td>
</tr>
<tr>
<td>SpecifyDataSourceField</td>
<td>N/A</td>
</tr>
<tr>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Note on Bound Reports:**

RDC versions 6 and 7 do not support the binding of a report to a Data Control or True DB Grid control to create an ad hoc report. This is an exclusive feature of the OCX control. The RDC uses Active Data for binding recordsets to existing report templates, which is much more flexible. See Active Data on page 16.

In Seagate Crystal Reports 8, the RDC lets developers create reports at runtime through code or an intuitive Report Expert (runtime licensing fees apply).