crystal decisions
A SEAGATE COMPANY

Crystal Reports

How Visual Basic Developers can migrate from the Crystal Report Engine Automation Server to the Report Designer Component

---

Overview

Since it was first included in Microsoft® Visual Basic®, Seagate Crystal Reports™ is the market-leading reporting tool with more than 4 million licenses shipped. It has kept pace with technological advancements by giving Visual Basic developers new ways to integrate reporting into database applications.

The purpose of this technical brief is to illustrate the benefits of using the newest technology—Version 8 of the Report Designer Component (RDC)—for integrating reporting functionality into Visual Basic applications, and to help you migrate applications from the Crystal Report Engine Automation server (CPEAUT) to the RDC to take advantage of the latest features within Seagate Crystal Reports. 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 CPEAUT.

Crystal Decisions will continue to focus its research and development efforts on the RDC. Developers who have created applications using the CPEAUT and plan to create new applications with the RDC will find this paper an ideal resource.

Contents

The Report Designer Component

The Report Designer Component (RDC) is an integrated and powerful solution for Visual Basic developers that quickly and easily integrates reporting into their database applications. It's an ActiveX designer object that packs the reporting power of Seagate Crystal Reports into a lightweight add-in for Visual Basic 5.0 or 6.0. Developers can open, design and customize reports within the Visual Basic IDE. Intuitive Report Experts make it flexible and efficient to connect to data and integrate powerful reports into applications. With hundreds of report properties, methods and events, 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. Reports can thus be packaged within the application's executable or stored outside the application in the traditional (.rpt) format.

RDC Architecture

Developers who aren't familiar with the RDC should note that it consists of three components. Together, these components enable developers to create, program, and preview, print, or export their reports:

- The **Main Engine** (craxdrt.dll)—known as the RDC runtime engine—is an extensive object model with hundreds of properties and methods for developers to use to program a report.
- The **Crystal Report Viewer** lets you preview reports onscreen for greater control and flexibility over the viewed report.
- The **Report Designer** is integrated tightly within Visual Basic 5.0 and 6.0, enabling developers to create, view, and modify reports in the Visual Basic IDE. Created specifically for Visual Basic developers, this component makes report design more intuitive. 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. This makes creating a report almost as easy as inserting a form.

Note that since the RDC has its own engine (craxdrt.dll), it does not need any links to the Crystal Reports Print Engine (CRPE), effectively acting as a stand-alone product.

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 to access the correct object or collection. See **Appendix B** for a diagram of the RDC Object Model.

The CPEAUT

The CPEAUT is still based on the original architecture of Seagate Crystal Reports 6, with a limited set of enhancements for versions 7 and 8. It is entirely based on the Crystal Report's print engine (CRPE) and technically is a wrapper around it, exposing only a subset of its functionality.

Code Comparison between the CPEAUT and RDC

The RDC is based on the current generation of Microsoft ActiveX technology. It's the method Visual Basic developers must use to take full advantage of the features within the Crystal Report Print Engine. Applications that are created using the CPEAUT will not be able to use the latest powerful Seagate Crystal Reports technology, especially the report creation API (new to the RDC version 8). If you're planning future releases or new applications, and you'd like to use the most powerful and flexible tool, consider the RDC.

Visual Basic developers can benefit from using the RDC with increased control over reports: flexible formatting like passing text to a text object; the latest Print Engine features such as mapping and multiple parameters; and most importantly, the ability to create, view and modify reports inside the Visual Basic IDE.

CPEAUT and RDC Sample Application Comparison

Following are two applications that provide similar functionality—the first is created using the CPEAUT, the second uses the RDC. The RDC example shows how to create a new application or convert an existing CPEAUT application. The RDC, with few exceptions, can duplicate any properties and methods set by the CPEAUT. Its object hierarchy is very similar to the CPEAUT, 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 Crystal Report Viewer for viewing reports.

Users of version 7 of the RDC (included in Seagate Crystal Reports 7) should select 'Crystal Report Designer Component Object Model' in the index of the Crystal Report Designer Online Help (crrdc.hlp). Users of version 8 of the RDC (included in Seagate Crystal Reports 8 Developer Edition) should select 'Report Designer Component Object Model Index' in the index of the Report Designer Component Online Help (crrdc.hlp).

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 CPEAUT application consists of a Form with three Command Buttons and a project reference to the Crystal Reports Engine Object Library.

Form Load:
- The Report is opened
- The location of the database in the main report is changed
- The parameter in the main report is set
- The subreport is opened
- The location of the database in the subreport is changed
- A string is passed to the formula field in the subreport

Command1
- The report is previewed to screen

Command2
- The printer is selected
- The report is printed

Command3
- The export options are set to export the report to a Rich Text Format
- The report is exported

A second form will be added when the application is created using the RDC. The Crystal Report Viewer is added to the second form for viewing the report.

CPEAUT Version:

Project | References:
- For version 7, reference the Crystal Report Engine 7 Object Library
- For version 8, reference the Crystal Report Engine 8 Object Library

```
Form1
(General – Declarations)


Dim CrystalApp As CRPEAuto.Application
Dim CrystalReport As CRPEAuto.Report
Dim CrystalSubreport As CRPEAuto.Report
Dim CrystaTable As CRPEAuto.DatabaseTable
Dim CrystaParameter As CRPEAuto.ParameterFieldDefinition
Dim CrystaFormula As CRPEAuto.FormulaFieldDefinition
Dim CrystalExportOptions As CRPEAuto.ExportOptions


Private Sub Form_Load()


'Create the application object, the "root" of all
automation objects
```

```
'in a CPEAUT application
Set CrystalApp = CreateObject("Crystal.CRPE.Application")

'Open the report
Set CrystalReport = CrystalApp.OpenReport(App.Path &
                    "\CPEAUT_to_RDC.rpt")

'Change the location of the database for each table in the
report
For Each CrystaTable In CrystalReport.Database.Tables
CrystaTable.Location = App.Path & "\xtreme.mdb"
             Next

'Pass the parameter value to the main report
'Assume the parameter is a string to be set to "Main Report
Param"
Set CrystaParameter = CrystalReport.Parameters.Item(1)
CrystaParameter.SetCurrentValue "Main Report Param",
crStringField

'Pass the selection formula to the main report
CrystalReport.RecordSelectionFormula = _
"{Customer.Last Year's Sales} < 50000.00"

'Open the subreport called "Sub1"
CrystalSubreport = CrystalReport.OpenSubreport("Sub1")

'Change the location of the database for each table in the
subreport
For Each CrystaTable In CrystalSubreport.Database.Tables
CrystaTable.Location = App.Path & "\xtreme.mdb"
             Next

'Pass the formula to the subreport
Set CrystalFormula = CrystalReport.FormulaFields
CrystalFormula.Text = "'Subreport Formula'"

End Sub

Private Sub Command1_Click()

'Preview the Report on a new window with the title "My
Report"
```

```
CrystalReport.Preview "My Report"


End Sub


Private Sub Command2_Click()


'Set the printer information
PrinterDriver$ = "HPPCL5MS.DRV"
PrinterName$ = "HP LaserJet 4m Plus"
PrinterPort$ = "\\Vanprt\v1-1mpls-ts"


CrystalReport.SelectPrinter PrinterDriver, PrinterName,
PrinterPort


'Print the report without prompting the user
CrystalReport.PrintOut False


End Sub


Private Sub Command3_Click()


'Set all the options to export the report to a Rich Text
Format file
Set CrystalExportOptions = CrystalReport.ExportOptions
CrystalExportOptions.FormatType = crEFTRichText
CrystalExportOptions.DestinationType = crEDTDiskFile
CrystalExportOptions.DiskFileName = App.Path &
"\CPEAUTExport.rtf"


'Export the report without prompting the user
CrystalReport.Export False


End Sub
```

RDC Version:

To migrate this application to the RDC, clear the reference to the Crystal Report Engine Object Library from the Project | References menu, in addition to the steps below:

Project | References

- For the RDC version 7, reference the Crystal Report 7 ActiveX Designer Runtime Library
- For the RDC version 8, reference the Crystal Report 8 ActiveX Designer Runtime Library

Project | Components
Crystal Report Report Viewer

Add a second form
Add the Crystal Report Viewer to Form2

For the most part the code is exactly the same as in the CPEAUT application. The three main differences are:

- The variables are declared as CRAXDRT instead of CPEAUT
- The report is previewed through the Crystal Report Viewer.
- The method to pass the parameter uses AddCurrentValue for the RDC versus SetCurrentValue for CPEAUT.

This will apply to most conversions from CPEAUT to RDC. The RDC object model is flatter than the CPEAUT. The steps to access some objects will vary between models. Refer to Appendix B 'Graphical Overview of the RDC Object Model'. The RDC contains additional methods and procedures and handles some functions differently than CPEAUT.

The RDC will open a standard Crystal Report (.RPT). The Report could have been Imported into or recreated in the RDC ActiveX Designer (.DSR).

For information on all the methods and procedures of the RDC and using a DSR users of version 7 of the RDC (included in Seagate Crystal Reports 7) should refer to the Crystal Report Designer Online Help (Crrdc.hlp). Users of version 8 of the RDC (included in Seagate Crystal Reports 8 Developer Edition) should refer to the Report Designer Component Online Help (Crrdc.hlp) and the Developer Online Help (Developr.hlp).

Form1:

```
(General - Declarations)

Dim CrystalApp As CRAXDRT.Application
Dim CrystalReport As CRAXDRT.Report
Dim CrystalSubreport As CRAXDRT.Report
Dim CrystalTable As CRAXDRT.DatabaseTable
Dim CrystalParameter As CRAXDRT.ParameterFieldDefinition
Dim CrystalFormula As CRAXDRT.FormulaFieldDefinition
Dim CrystalExportOptions As CRAXDRT.ExportOptions

Private Sub Form_Load()
'Create the application object, the "root" of all objects
'in a RDC application
Set CrystalApp = CreateObject("CrystalRuntime.Application")
```

```
'Open the report
Set CrystalReport = CrystalApp.OpenReport(App.Path &
                    "\CPEAUT_to_RDC.rpt")

'Change the location of the database for each table in the
report
For Each CrystaTable In CrystalReport.Database.Tables
CrystaTable.Location = App.Path & "\xtreme.mdb"
             Next

'Pass the parameter value to the main report
'Assume the parameter is a string to be set to "Main Report
Param"
Set CrystaParameter = CrystalReport.Parameters.Item(1)
CrystaParameter.AddCurrentValue "Main Report Param"

'Pass the selection formula to the main report
CrystalReport.RecordSelectionFormula = _
"{Customer.Last Year's Sales} < 50000.00"

'Open the subreport called "Sub1"
CrystalSubreport = CrystalReport.OpenSubreport("Sub1")

'Change the location of the database for each table in the
subreport
For Each CrystaTable In CrystalSubreport.Database.Tables
CrystaTable.Location = App.Path & "\xtreme.mdb"
             Next

'Pass the formula to the subreport
Set CrystalFormula = CrystalReport.FormulaFields
CrystalFormula.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.

CrystalReport.SelectPrinter "HPPCL5MS.DRV", "HP LaserJet 4m
Plus", "\\Vanprt\v1-1mpls-ts"


'Print the Report without prompting user

CrystalReport.PrintOut False


End Sub


Private Sub Command3_Click()

'Set all the options to export the report to a Rich Text
Format file

Set CrystalExportOptions = CrystalReport.ExportOptions
CrystalExportOptions.FormatType = crEFTRichText

CrystalExportOptions.DestinationType = crEDTDiskFile

CrystalExportOptions.DiskFileName = App.Path &
"\CPEAUTXExport.rtf"


'Export the report without prompting the user

CrystalReport.Export False


End Sub


Form2:


Private Sub Form_Load()


'Set the Report source for the Crystal Report Viewer to the
Report

CRViewer1.ReportSource = Form1.CrystalReport


'View the Report

CRViewer1.ViewReport


End Sub


Private Sub Form_Resize()


'This code resizes the Report Viewer control to Form2's
dimensions

CRViewer1.Top = 0
```

```
CRViewer1.Left = 0
CRViewer1.Height = ScaleHeight
CRViewer1.Width = ScaleWidth


End Sub
```

Features Exclusive to the RDC

The RDC offers many advanced features that are not available in the CPEAUT. As Crystal Decisions releases future versions of Seagate Crystal Reports, new features will only be available in the RDC.

Features exclusive to the RDC include:

- **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** - customize the Export Button. In version 7 of the RDC, this component was called the Crystal Smart Viewer.
- **Format at runtime** - pass text to a TextObject, format a field at runtime, format a field in a section event, load a picture in a section event.
- **Report creation at runtime**[1] – Design and format a report at runtime through a report Expert or code (runtime license applies)
- **Unbound fields**[1] – Bind fields at runtime.
- **Change the runtime location of an OLE object**[1] – Change the location of any OLE object through the Section Format event.

1. Printer options

Version 8 of the RDC is now 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.

Set PaperSource and PrinterDuplex:

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


Private Sub Form_Load()
'Set the paper source to the lower bin
Report.PaperSource = crPRBinLower
'Set duplex printing to horizontal
```

[1] New in Seagate Crystal Reports 8

```
Report.PrinterDuplex = crPRDPHorizontal

'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
```

2. Enhanced parameters

The ability to take full advantage of multiple, ranged, and multiple-ranged parameters is only available in the RDC in versions 7 and 8 of Seagate Crystal Reports

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 2, 8000
Report.ParameterFields.Item(1).SetNthDefaultValue 3, 12000
```

```
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
Report.ParameterFields.Item(1).AddCurrentRange 1000, 10000,
_ crRangeIncludeLowerBound + crRangeIncludeUpperBound

'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.ParameterFields.Item(1).AddCurrentRange 20000,
25000, _ crRangeIncludeLowerBound +
crRangeIncludeUpperBound
```

3. Report Viewer

The Report Viewer is an ActiveX control invoked by the application to present a report to the end user. Users of the viewer can navigate and analyze reports, print the report, export the report 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:

Events are not exclusive to the RDC. The RDC handles all window events through the Crystal Report Viewer. 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
Report.ExportOptions.DiskFileName = "C:\ExportReport.rtf"

'Export without prompting user
Report.Export False
End Sub
```

4. 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 pictures at runtime, and setting properties based on a field's value. The RDC uses reports created in the Seagate 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 pictures 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"
```

5. 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
Report.Field1.Font.Bold = True
```

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 a picture 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
```

6. Report Creation at Runtime

Developers can choose between a Report Expert or code to integrate runtime report creation capabilities into their applications. This feature of the RDC is only available in Seagate Crystal Reports 8, Developer Edition.

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
Set crxReport = crxApplication.NewReport


'Add the Customer table from xtreme.mdb
crxReport.Database.Tables.Add App.Path & "\xtreme.mdb",
"Customer"


'Set crxReport to the Crystal Report Wizard
Set crxWizard.CrystalReport = crxReport


'Display the Crystal Report Wizard
crxWizard.DisplayReportWizard


End Sub
```

Using the Report Creation API:

The Seagate Crystal Reports 8, Developer Edition exposes the Report Creation API to allow 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

Set crxReport = crxApplication.NewReport


'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 http://www.crystaldecisions.com/products/crystalreports/licensing.

7. Unbound Fields

Unbound fields are fields of a specific data type, which 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. This feature of the RDC is only available in Seagate Crystal Reports 8.

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
seperators.

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}"
Report.UnboundString1.SetUnboundFieldSource
"{Customer.Customer Name}"

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

8. Change the Runtime Location of an OLE Object

One of the top features of RDC versions 7 was the ability to load a picture into an OLE object at runtime through the section format event. Version 8 of the RDC takes this a step further by allowing 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
```

Active Data:

One of the cornerstones of the RDC is the ability to report off Active Data. Active Data is the ability to set the data source 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.

The Active Data Driver provides a user-friendly interface for the programmer to choose a source from which to create the report template. This includes using an existing ODBC data source, OLEDB connection string (Version 7 only), browsing to a PC data source (*.mdb, *.dbf, etc), or a Data Definition File (*.ttx). Once the report is created a recordset can be passed to the report at runtime.

```
‘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 Smart Viewer
CRViewer1.ReportSource = Report
'View the report
CRViewer1.ViewReport

End Sub
```

Summary

The RDC represents the latest in ActiveX technology and provides the following advantages over the CPEAUT:

- Integrates directly into the Visual Basic IDE
- Allows you to create, view, and modify reports using Reports Experts and familiar Visual Basic code
- Exposes all Print Engine features and provides the greatest number of events and objects to write code to
- Better performance from its dual interface component with no wrapper around the Print Engine
- Take advantage of code completion features that are easy to use in the Visual Basic editor
- Fully compatible with Microsoft Visual Basic 5.0 and 6.0
- Provides access to the Report Creation API (not discussed in this paper) in version 8

Two versions of the RDC are currently available. Version 7 is included in Seagate Crystal Reports 7 Professional and includes support for Visual Basic features including the Data Environment. Version 8 is included in Seagate Crystal Reports 8 Developer edition and includes enhanced features and access to the Report Creation API.

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.

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. These properties are 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 SetLogonInfo property of the DatabaseTable object is needed to connect to the server. The following code will demonstrate three methods to get to this property. Example1 will set a variable for each individual object from the Report object to the DatabaseTable object before the SetLogonInfo property is set.

Example2 will only set the variable for the DatabaseTable object before the SetLogonInfo property is set. Example3 will set the SetLogonInfo 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)


'Connect using the SetLogonInfo property of
crxDatabaseTable

crxDatabaseTable.SetLogOnInfo "DSN", "Database", "UserID',
_ "Password
```

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)

'Connect using the SetLogonInfo property of the
crxDatabaseTable Object.
```

```
crxDatabaseTable.SetLogOnInfo "DSN", "Database", "UserID',
_ "Password
```

Example 3:

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

Report.Database.Tables.Item(1).SetLogOnInfo "DSN",
"Database", "UserID', _ "Password
```

Setting a Property or Method for all Objects in a Collection

A collection may contain many objects. 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 set the SetLogonInfo property for all objects in the DatabaseTables collection. Example1 will set each DatabaseTable object individually. Example2 will use the Count property of the DatabaseTables collection to create a For Loop and cycle through all DatabaseTable objects collected. Example3 will use the Visual Basic 'For Each' command to access each DatabaseTable object in the DatabaseTables collection.

Example1:

```
Dim crxDatabaseTable as craxdrt.DatabaseTable


'Set crxDatabaseTable to the first DatabaseTable in the
DatabaseTables 'collection.

Set crxDatabaseTable = Report.Database.Tables.Item(1)

'Connect using the SetLogonInfo property of
crxDatabaseTable

crxDatabaseTable.SetLogOnInfo "DSN", "Database", _

"UserID',  "Password"

'Set crxDatabaseTable to the second DatabaseTable in the
DatabaseTables 'collection.

Set crxDatabaseTable = Report.Database.Tables.Item(2)

'Connect using the SetLogonInfo property of
crxDatabaseTable

crxDatabaseTable.SetLogOnInfo "DSN", "Database", _

"UserID',  "Password"

'Repeat until all DatabaseTables are set.
```

Example2:

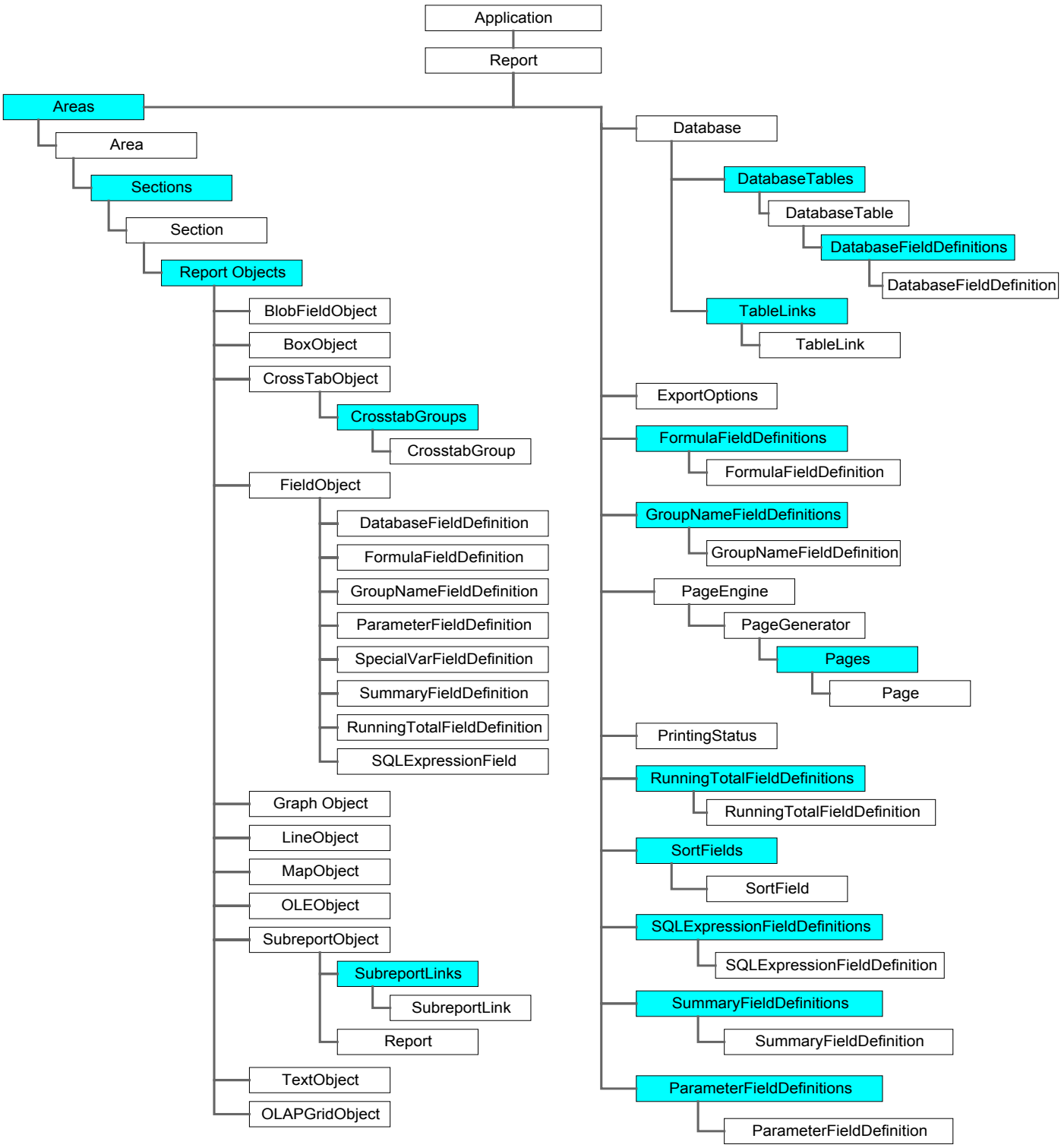
```
Dim crxDatabaseTable as craxdrt.DatabaseTable

Dim nTable as Integer
```

```
                For nTable = 1 to Report.Database.Tables.Count
'Set crxDatabaseTable to the nth DatabaseTable in the
'DatabaseTables collection.


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


'Connect using the SetLogonInfo property of the
'crxDatabaseTable 'Object.
crxDatabaseTable.SetLogOnInfo "DSN", "Database", _
"UserID',  "Password"
Next nTable
```

Example3:

```
'Set crxDatabaseTable to each DatabaseTable in the
DatabaseTables collection.
'For Each crxDatabaseTable in Report.Database.Tables
'Connect using the SetLogonInfo property of the
crxDatabaseTable 'Object.
crxDatabaseTable.SetLogOnInfo "DSN", "Database", _
"UserID',  "Password"
Next crxDatabaseTable
```

Appendix B: Graphical Overview of the RDC Object Model



Contacting Crystal Decisions for Technical Support

Along with this document, and the *Crystal Reports User's Guide*, we recommend that you visit our Technical Support web site for further resources and sample files. For further assistance, visit us at the web sites below.

Technical Support web site:

http://support.crystaldecisions.com/homepage/

Answers By Email Support:

http://support.crystaldecisions.com/support/answers.asp

Phone Support:

Tel: (604) 669-8379