Seagate Crystal Reports™ 8 Developer's Guide

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

C O N T E N T S

1 What's New for Developers

What you will find in this chapter

OVERVIEW

As developers, you represent a major group of Seagate Crystal Reports users. You are important to us. You have asked us for many specific additions to the product and we have listened. Version 8 of Seagate Crystal Reports is full of new functionality and new tools to help you get your job done with fewer headaches and with less code.

This chapter highlights the key changes we've made for developers, and it points you to further information. For a more complete discussion of some of the more major changes, see *A Closer Look at Some of the New Features, Page 22*.

REPORT INTEGRATION CONTROLS FOR VISUAL INTERDEV

The Report Integration controls allow Web developers using Microsoft Visual InterDev 6.0 to write Web applications that generate Crystal reports hosted on either a Web Reports Server or an ASP server. End users of these applications would then view these reports using a Report Viewer running on a browser.

You control such things as logging on to database servers, prompting of parameters and specifiying selection formulas in these reports. You allow your users to view reports using either the Report Viewer for Java or the Report Viewer for ActiveX. You can also decide which properties to enable for the end user within the Report Viewer (printing or exporting, for example).

SERVING UP REPORTS FROM LOTUS DOMINO APPLICATIONS

Now you can integrate your reports into applications that you develop using Lotus Domino. You can use any of several tools provided by Seagate Crystal Reports including the Automation Server and the ActiveX control. An in depth discussion of integrating reports with Lotus Domino applications is provided in Chapter 10, *Working with Visual C++, Visual InterDev and Lotus Domino, Page 125.*

ENHANCEMENTS TO THE CRYSTAL REPORT PRINT ENGINE API

There have been many enhancements to the Crystal Report Print Engine (CRPE) API. Many new API structures and functions have been introduced and many existing ones have been enhanced, including the following:

Version 8 Enhancements

There are many enhancements to the API that correspond to enhancements in Version 8 of Seagate Crystal Reports. For a more detailed description of the enhancements that follow, refer to the documentation for the related functionality in the *Seagate Crystal Reports User's Guide*. For developer reference information on each of the new and enhanced calls, see the Developer's help or the *Seagate Crystal Reports Technical Reference Guide*.

Launch Seagate Analysis

You may now have users click a button on the preview window toolbar to launch Seagate Analysis and open the current report.

A new event, LaunchSeagateAnalysisEvent, has been added. If you enable the event, and the user clicks the Launch Seagate Analysis button, the LaunchSeagateAnalysisEvent will be fired. The user will be prompted to save the file and Seagate Analysis will be launched with the current report open.

There are a number of other enhancements in the CRPE API that complement those in corresponding components of the Crystal Report Designer.

Accounting Conventions

To support conventions used in accounting, you may now control how the negative symbol, currency symbol and zero value are displayed and reverse the sign when displaying numbers in accounting reports.

The CRPE API has also been enhanced to complement these changes in the Crystal Report Designer. You may use:

- PEGetObjectFormat, PESetObjectFormat and PESetNthEmbeddedFieldFormat with objectFormatName set to:
  - PE_OFN_REVERSE_SIGN_FOR_DISPLAY
  - PE_OFN_USE_ACCOUNTING_FORMAT
  - PE_OFN_SHOW_ZERO_VALUE_AS
- PEGetObjectFormatFormula and PESetObjectFormatFormula with formulaName parameter set to PE_FFN_REVERSE_SIGN_FOR_DISPLAY

Basic and Crystal Syntax Support

Since the Seagate Crystal Reports formula language has been enhanced to support a Basic-like syntax as well as the Crystal syntax, existing Formula API calls have been modified accordingly.

In all the Formula calls now, SET uses the syntax specified by the user and GET sets a flag that the user can later retrieve.

Two new API calls, PESetFormulaSyntax and PEGetFormulaSyntax, have also been added.

Charting Enhancements

The following are several enhancements for charting:

Automatic Scaling for Axes

In the CRPE API, three new elements are now supported in the structure, PEGraphAxisInfo:

- dataAxisYAutoScale
- dataAxisY2AutoScale
- seriesAxisAutoScale

You may use the CRPE API calls, PEGetGraphAxisInfo and PESetGraphAxisInfo, to obtain or set automatic scaling for the graph axes.

Default Titles

The Crystal Report Designer, by default, now creates title, subtitle, footnote, X-axis title, Y-axis title and Z-axis title. In the CRPE API, you can use PEGetGraphTextDefaultOption and PESetGraphTextDefaultOption to manipulate these items. To use these calls, you will need to specify a PE_GTT_* constant that corresponds to each of the titles, and set useDefault as TRUE or FALSE.

Legend Layout

In the CRPE API, you may now set the legendLayout element in PEGraphOptionInfo structure to one of the PE_GLL_* constants. You may also use the CRPE API calls, PEGetGraphOptionInfo and PESetGraphOptionInfo, to get or set legend layout options.

Fractional Point Size

For fonts, there is now support for fractional point sizes including the 10.5 point font size commonly used in Japanese. A new twipSize element has been added to the structure PEFontColorInfo. Related functions and methods now return or apply the twipSize element if the pointSize element is zero. The functions affected are:

- PEGetObjectFontColor
- PESetObjectFontColor
- PEGetTextFontColor
- PESetTextFontColor
- PEGetGraphFontInfo
- PESetGraphFontInfo

Hierarchical Grouping

In the Crystal Report Designer, you may now use hierarchical grouping to arrange data in a report to show hierarchical relationships in your data.

To support hierarchical grouping, the PEGroupOptions structure and the PEGetGroupOptions and PESetGroupOptions calls in the CRPE API have been modified.

Hyperlinks & On-demand Subreports

The CRPE API has been enhanced to support hyperlinks and on-demand subreports.

New elements have been added to the PETrackCursorInfo structure to support these enhancements:

- short ondemandSubreportCursor – this cursor can be set to display over on-demand subreports when drilldown for the window is enabled; the default is PE_TC_MAGNIFY_CURSOR.
- short hyperlinkCursor – this cursor can be set to display over any report object that contains hyperlink text; the default is PE_TC_HAND_CURSOR.
- new cursor types:
  - PE_TC_BACKGROUND_PROCESS_CURSOR
  - PE_TC_GRAB_HAND_CURSOR
  - PE_TC_ZOOM_IN_CURSOR
  - PE_TC_REPORT_SECTION_CURSOR
  - PE_TC_HAND_CURSOR

SCR 7 or Earlier Enhancements

There are also many enhancements to the API that correspond to features in Seagate Crystal Reports 7 or earlier versions:

OLAP Grid Objects

OLAP Grid objects have been exposed in the CRPE API. You may now specify PE_OI_OLAPGRIDOBJECT in the PEObjectInfo structure, and use it in PEGetObjectInfo and PESetObjectInfo calls.

Secondary Summary Field

The CRPE API has several enhancement to support secondary summary fields:

- Two new function calls have been added: PESetSummaryFieldInfo and PESetNthSummaryInfoInCrossTabObject.
- New elements have been added to the structures PEFieldDefinitionInfo, PESummaryFieldInfo and PECrossTabSummaryFieldInfo.
- The PEGethNthSummaryInfoInCrossTabObject function has been modified.

Saving in Seagate Crystal Reports 7 Format

The report file format has been changed in Seagate Crystal Reports 8. Consequently, there is an option in the Crystal Report Designer to allow you to save a report in either the Version 7 or Version 8 format. Similarly, in the CRPE API, you can use a new call, PESavePrintJobAs, and specify:

- PE_SAVEAS_SCRFORMAT_DEFAULT for Version 8 format, or
- PE_SAVEAS_SCRFORMAT_V7 for Version 7 format.

Text Formatting

To complement the enhancements in the Crystal Report Designer on line spacing, character spacing, text rotation and indentation for text objects and string field objects, the CRPE API has been enhanced in the following ways:

- **For text interpretation**
  You can now specify whether the program should override the default format with plain text interpretation:
  - use PEGetObjectFormat, PESetObjectFormat, PEGetObjectFormatFormula and PESetObjectFormatFormula with the objectFormatName parameter being set as PE_OFN_TEXT_INTERPRETATION.

- **For line spacing**
  You can now specify whether line spacing should be a multiple of the size of the font or an exact number of points.

  Use PEGetObjectFormat and PESetObjectFormat with objectFormatName being set as:

  — PE_OFN_LINE_SPACING_TYPE—specifies whether line spacing is a multiple of the size of the font, or an exact number of points, or

  — PE_OFN_LINE_SPACING—specifies that the line spacing is an exact number of points.

- **For character spacing**
  Use PEGetObjectFormat and PESetObjectFormat with ObjectFormatName being set as PE_OFN_CHARACTER_SPACING.

- **For text rotation**
  Use PEGetObjectFormat and PESetObjectFormat with objectFormatName being set as PE_OFN_TEXT_ROTATION.

- **For line indentation**
  You can now specify the kind of indentation a line of text should have. To do this, you use PEGetObjectFormat and PESetObjectFormat and set the objectFormatName parameter to:

  — PE_OFN_FIRST_LINE_INDENT for first line indentation,

  — PE_OFN_LEFT_INDENT for left indentation, or

  — PE_OFN_RIGHT_INDENT for right indentation.

WYSIWYG Support

You can now give your users the chance to view and print reports in either WYSIWYG or non-WYSIWYG mode.

WYSIWYG mode uses the printer driver settings for previewing the report. What you see on screen approximates the way the report will print.

Non-WYSIWYG mode uses the screen driver setting for previewing the report. The printed report may differ from the report as seen on screen.

To support this feature, the structure PEReportOptions has been modified and a new element, wysiwygMode, has been added.

ENHANCEMENTS TO REPORT DESIGNER COMPONENT

The Report Designer Component has seen the most changes of all, including the following:

CRPE API

More of the CRPE API functionality is now available through the RDC.

The following new interfaces have been added:

- ITableLinks
- ITableLink
- IFieldDefinitions
- IObjectSummaryFieldDefinitions
- ICrossTabGroups
- ICrossTabGroup
- ISubreportLinks
- ISubreportLink

The following new methods have been added to existing interfaces:

- IDatabase
- IDatabaseTables
- IDatabaseTable
- IFormulaFieldDefinitions
- ParameterFieldDefinitions
- ISQLExpressionFieldDefinitions
- IRunningTotalFieldDefinitions
- ISummaryFieldDefinition
- IRunningTotalFieldDefinition
- IReport
- ISections
- ISection
- IGraphObject
- ICrossTabObject

Report Variables

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

To prevent these problems, the RDC now enables you to declare Report Variables. Report Variables are Crystal variables that you can use in your VB code. Report Variables "understand" what's going on in the report and they increment appropriately. This enables you to do more of your coding than ever in VB and still get the results you expect. For more information, see *Using Report Variables, Page 22*.

To support report variables, the methods AddReportVariable, GetReportVariableValue and SetReportVariableValue have been added to the IReport interface.

Set OLE Object Location

You can now set the location of an OLE object at runtime using the new method, SetOleLocation, under the ICROleObject interface. This also enables you to change an OLE object to point to different files at run time including image files.

The existing method, SetFormattedPicture, is still supported. However, if you use both SetFormattedPicture and SetOleLocation, SetFormattedPicture overrides SetOleLocation.

Also note that if an OLE object is in the Detail section and you call SetOleLocation in the OnFormat event for every detail record, this will affect your performance significantly.

Save Preview Picture

In the Crystal Report Designer, you now have the option to save a snapshot of the first page of a report for previewing in the File Open dialog box. This capability is now supported in the RDC.

Two methods have been added to the Report Designer Component runtime, under the IReport interface: get_SavePreviewPicture and put_SavePreviewPicture. You can use get_SavePreviewPicture to get the property and use put_SavePreviewPicture to set the property.

> ***Note: You should be very careful to enable this flag at design time. If you don't and if the report contains thumbnail pictures, it can affect the performance when saving or loading the report.***

Charts

The following charting enhancements are carried through the Crystal Report Designer and the Report Designer Component:

Automatically Scale Data Axis

Three new runtime properties have been added under the class GraphObject: EnableAutoScaleDataAxis, EnableAutoScaleData2Axis and EnableAutoScaleSeriesAxis.

Legend Enhancements

In the Report Designer Component runtime, the new enumerator type CRPieLegendLayout has been added, and the new property LegendLayout has been added under the class GraphObject.

Chart on All Records

Please refer to the *Seagate Crystal Reports User Guide* for a detailed description of this feature.

Chart on Running Totals

Please refer to the *Seagate Crystal Reports User Guide* for a detailed description of this feature.

Default Titles for Charts

Previously, when you used the Report Expert to create a chart, the expert gave it a title unless you specified one. Now the expert supplies the following titles by default: title, subtitle, footnote, X-axis title, Y-axis title and Z-axis title.

To support these enhancements, these new properties have been added to the GraphObject class for the Report Designer Component runtime:

- IsTitleByDefault
- IsSubTitleByDefault
- IsFootnoteByDefault
- IsGroupTitleByDefault
- IsSeriesTitleByDefault

- IsXAxisTitleByDefault
- IsYAxisTitleByDefault
- IsZAxisTitleByDefault

Font Change for Chart Elements

You may now change the font used for chart elements. In the Report Designer Component runtime, properties have been added to the GraphObject class to change the font for these chart elements: title, subtitle, footnote, group title, data title, series title, legend, group labels, data labels and series labels.

New Formats for Data Points and Data Axes in Charts

Four new formats have been added to the Chart Expert to format data point values and data axis values: 1K, 1M, $1K and $1M.

In the Report Designer Component runtime, the following properties have been added to the GraphObject class to support this:

- DataAxisNumberFormat
- Data2AxisNumberFormat
- SeriesAxisNumberFormat

Creating Chart Based on a Cross-Tab

You may now create a chart based on a cross-tab object. A new method, AddGraphObject, has been added with the parameter pCrossTabObject. pCrossTabObject is optional, is a VARIANT type, and has to be a CrossTabObject. (For crGroupGraph and crDetailGraph, this parameter is ignored.)

Creating Reports at runtime using code

The RDC now includes Report Creation API methods. These methods enable you to create ad hoc reports at runtime using nothing but code. The methods can be used within Visual Basic, Microsoft Office, or in any environment from which you can access an automation server. Because these methods now enable Seagate Crystal Reports' Designer functionality in your runtime applications, they require separate licensing.

Cross-Tab reports

Cross-Tab Formatting

The enhancements to the Crystal Report Designer for formatting cross-tabs (For example, predefined grid styles, formatting separate grid lines) are now also available in the Report Designer Component at design time.

Selecting & Formatting Cells in Cross-Tabs

You may now select individual cells in a cross-tab, or select multiple cells in a cross-tab, to format and re-size them. This is a design-time only feature.

Creating Chart Based on a Cross-Tab

You may now create a chart based on a cross-tab object. The new parameter, pCrossTabObject, has been added to the AddGraphObject method. pCrossTabObject is optional, is a VARIANT type, and has to be a member of ICrossTabObject. (For crGroupGraph and crDetailGraph, this parameter is ignored.)

ADO and OLEDB Tables

Two new methods have been added under the IDatabase interface to give you more control when working with ADO and OLEDB data:

- AddADOCommand (connection, command)
- AddOLEDBSource (connection, tableName)

These two methods enable you to add an ADO and OLE DB data source, save the report, and then refresh the report.

The methods from earlier versions, SetDataSource and Add (under IdatabaseTables) allow you to set a run time ADO record set as a data source, but the report saved will not be refreshed.

Data Explorer

The Report Designer Component has been enhanced to use the Data Explorer when you:

- use a Crystal Report Designer driver to create a report through the Report Expert,
- perform a Set Location, or
- perform an Add Database.

When you create a report or do a set location, you now use the new Data Explorer to manage your database connections. The Data Explorer provides an integrated tree view of all data sources available to you.

For more information on the Data Explorer, please refer to *Seagate Crystal Reports User's Guide.*

Field Explorer

The Field Explorer has been enhanced. It now includes an Unbound Fields node.

Unbound Fields

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

This functionality makes it easy to build reporting into n-tier applications where you have on-demand recordsets and non-continuous data connections. It also makes it easy to build a single report that can work with a variety of recordsets, making development and maintenance more efficient.

Unbound fields allow for great flexibility, since you can easily change the data source for an unbound field at runtime.

- In the Visual Basic RDC design time environment, you can easily add unbound fields to your report by dragging and dropping any of the unbound field types from the Designer's treeview onto the report.
- In the Report Designer Component runtime, you can add an unbound field object to a section using the new function AddUnboundFieldObject under the Section class.
- To bind the unbound field to a database field at runtime, you can use the new method SetUnboundFieldSource.
- Alternatively, you may use automatic unbound field binding, based on either the field name or the field name and value type of the field in the data source. To do so, you would use the new method AutoSetUnboundFieldSource, and specify one of the constants crBMTName or crBMTNameAndValue for CRBindingMatchType.

Select Distinct Records

You may now direct the RDC to select only unique records from a data source. In the IReport interface, EnableSelectDistinctRecords has been added to allow you to get or set this option based on user input.

Selecting & Editing Embedded Fields

If a database field is embedded in a text object, you can now select and edit the embedded objectfield as if it were not embedded.

Report Object Events

BeforeFormatPage & EndFormatPage Events

The BeforeFormattingPage and EndFormattingPage events are fired before and after formatting a page respectively. These events enable you to make a variety of page-specific changes to your report.

You could, for example, reset Report Variables that you were using to calculate a subtotal on a page-by-page basis. Another possibility is to control the report formatting information as desired before the page is formatted.

> ***Note: There are a number of properties and methods you cannot use in these events. For example, you cannot use the "SetText" method in BeforeFormatPage or EndFormatPage or you will get a runtime error. When the events are fired, the report is in Active formatting mode while the sections are in inactive formatting mode. Thus, you can use properties and methods that affect the report object, but you can't use properties and methods that affect anything in the sections of the report.***

FieldMapping Event

The FieldMapping event is fired if a database is changed while verifying database. Field mapping enables you to change the name of a database field that a report field points to if that field no longer exists or has been renamed in the data source. Verifying the database or using SetLocation will trigger the FieldMapping event.

> ***Note: This event is fired only when the fieldmapping type is not set to automatic. If it is set to automatic, the program will display a field mapping dialog box instead.***

NoData Event

The NoData event is fired when the data set in the report is empty. It enables you to stop the processing and / or invoke error handling.

Note: This event is fired only for the main report, even if there is a subreport.

Formatting

Text Formatting

To complement the enhancements in the Crystal Report Designer on line spacing, character spacing, text rotation and indentation for text objects and string field objects, the Report Designer Component runtime has also been enhanced in the following ways.

In the ITextObject interface for text objects, and the IFieldObject interface for string field objects, these properties and methods have been added:

- CharacterSpacing (read / write)
- LineSpacing (read)
- LineSpacingType (read)
- SetLineSpacing (method)
- TextRotationAngle (read / write)
- FirstLineIndent (read / write)
  When reading, this method gets first line indent of the first paragraph; when writing, it sets first line indent for all the paragraphs.
- LeftIndent (read / write)
  When reading, this method gets the left indent of the first paragraph; when writing, it sets the left indent for all the paragraphs.
- RightIndent (read / write)
  When reading , this method gets the right indent of the first paragraph; when writing, it sets the right indent for all the paragraphs.
- TextFormat (read / write)
  For text objects only.

Fractional Point Font Support

You may now specify a fractional point size for text objects and database fields, including the 10.5 point font commonly used in Japanese. To do so, type in a fractional point size in the Font tab of the Format Editor that you use to format a selected text object or database field.

Note: VB users cannot use this feature. While our own dialog box now supports fractional point sizes, Visual Basic does not support such sizes.

Accounting Conventions

To support conventions used in accounting, you may now control how the negative symbol, currency symbol and zero value are displayed, and reverse the sign when displaying numbers in accounting reports. To implement these conventions, the following properties have been added to the Report Designer Component runtime FieldObject class: UseReverseSign and ZeroValueString.

Formulas

Basic Syntax

The formula language in Seagate Crystal Reports 8 has been extensively enhanced and augmented with the addition of a new Visual Basic-like language called Basic syntax. It is an alternative language to the formula language. Most new functions, operators, and control structures have been added to both Crystal and Basic syntaxes, and most existing Crystal capabilities have been duplicated in Basic as well. While existing Seagate Crystal Reports users do not have to learn this new syntax to utilize the extra capabilities, it is intended to be easy to learn and use for developers already familiar with Visual Basic.

Apart from the new functions, operators and control structures available in both syntaxes, Visual Basic developers who distribute applications can benefit from the fact that Basic formulas are portable and do not require any additional DLL's to run.

You must use a single syntax within a given formula, but you may use formulas with different syntaxes in the same report. When you enter a formula (other than a selection or search formula - see note below) in the Formula Editor, you can choose either Crystal or Basic syntax as the default syntax for the formula.

New functions available to both syntaxes include:

- financial functions
- mathematics functions
- programming shortcut functions
- type conversion functions
- date and time functions
- array and string functions
- percentage functions

New operators available to both syntaxes include:

- integer division
- exponentiation
- eqv
- imp
- xor
- mod

New control structures available to both syntaxes include:

- for statement
- do statement
- select … case statement

For a more detailed description of these functions, operators and control structures, please refer to the Developer's Help.

> ***Note: The two syntaxes are fundamentally equivalent, with one exception: When you create or modify search or selection formulas using the Select Expert, the Search Expert, or the Formula Editor, you can only use Crystal syntax.***

FormulaSyntax and LastGetFormulaSyntax

Two new properties have been added to the Report class for working with these two languages: FormulaSyntax and LastGetFormulaSyntax.

- You can specify the syntax that you want to use before you create a formula via the FormulaSyntax property.
- After getting formula text, you can use the property LastGetFormulaSyntax to determine the syntax used for the formula.

Grouping

Hierarchical Reporting

You may now use hierarchical grouping to arrange data in a report to show hierarchical relationships. You still keep all the elements of the hierarchy in the same table, but you can now display the relationships between the elements. This can be useful, for example, when reporting off a Human Resources database, to show the organizational hierarchy within a company.

In the Report Designer Component runtime, the following properties have been added to implement this grouping under the Area class:

- EnableHierarchicalGroupSorting
- ParentIDField
- InstanceIDField
- GroupIndent

You can use the InstanceIDField and ParentIDField properties to organize the group sorting tree, use the GroupIndent property to design the layout, and use the EnableHierarchicalGroupSorting property to enable or disable hierarchical group sorting.

At design time you may select hierarchical grouping for your report in the Report Designer Component, by right-clicking and choosing Report | Hierarchical Grouping Options. This will invoke the Change Hierarchical Options dialog box.

For more information and an example of how to use this feature, please refer to the *Seagate Crystal Reports User's Guide*.

IDE

Toolbars

As in the Crystal Report Designer, toolbars now have an updated, flattened look, and can be moved. And, in particular to the Report Designer Component, the toolbars can also be re-sized to fit your needs.

WYSIWYG Support

At design time, you can turn on or off WYSIWYG support for the current report using the Report Options dialog box. For more information on WYSIWYG support in the Crystal Report Designer, please refer to *Seagate Crystal Reports User's Guide.*

API Enhancements from Seagate Crystal Reports 7 or earlier

Those enhancements that correspond with Seagate Crystal Reports 7 or earlier versions include the following:

Tool Tips

There are now tool tips for objects in the RDC design time that display the data type, object name, and field name for the object wherever applicable.

Show Object Names for Fields

In the RDC design time, you may now choose to display the object name for a field (the Name property) as it appears in the VB Properties Window. You can set this option on the Layout tab of the Default Settings dialog box.

MTS Support

The RDC runtime supports the apartment model (apartment threading) and can thus run under Microsoft Transaction Server (MTS). It does not support state free status, however, which means it cannot utilize certain MTS functionality.

When you install a component within MTS, MTS allocates a Transaction Support property that defines the way the component will behave. The property has four settings.

- Requires a transaction
- Requires a new transaction
- Supports transactions
- Does not support transactions

Because the RDC does not support state free status, when you install it to work within MTS, make certain you choose the "Does not support transactions" setting. This makes certain that the component will never attempt to run within a transaction. It will always run outside the transactions.

OLAP Grid Formatting

You may now import a report with an OLAP grid, select the grid and format it. When you right-click the grid, you can select Cut, Copy, or Format from the shortcut menu. If you choose format, the program invokes the Format Editor with the Common and Border tabs.

Exporting to Excel 7.0 and 8.0 Formats

You may now export to Excel 7.0, 7.0 Extended, 8.0 and 8.0 Extended formats from the Report Designer Component runtime.

Now, under the IExportOptions interface, there are four new formats:

- crEFTExcel70
- crEFTExcel70Tabular
- crEFTExcel80
- crEFTExcel80Tabular

Also, there are six new Excel format option properties:

- ExcelTabHasColumnHeadings
- ExcelUseWorksheetFunctions
- ExcelConstantColumnWidth
- ExcelAreaType
- ExcelAreaGroupNumber
- ExcelUseTabularFormat

You can also export now to PDF and HTML 4.0 formats.

Save As Version 7 Format

The report file format has been changed in Seagate Crystal Reports 8.

- In the Report Designer Component design time, you now have the option of saving the report in Version 7 or Version 8.
- In the Report Designer Component runtime, there is a new method under the IReport interface, SaveAs, that allows you to specify in which file format (Version 7 or Version 8) you would like to save your report.

The previously used method Save is now hidden.

Note: Not all Version 8 reports can be saved in Version 7 format and still work properly. Reports containing the following features will not work properly in Version 7 format:

- ***Formulas using the new Basic syntax***
- ***Formulas using new formula functionsReports created off OLE DB data source, especially if they use the extended style***
- ***Hierarchical grouping***
- ***Percentage summary fields***

Optimized Group Tree Creation & Deletion

Group tree creation and deletion at runtime have been optimized. Now you can begin viewing the first page of the report or navigating the group tree before the entire report and the entire group tree have been loaded.

Printing

Setting Printer Duplex and Paper Source

At runtime, you may set the printer duplex and paper source for the current report. There are two new properties that can be set and retrieved: PrinterDuplex and PaperSource. In combination with the existing properties PaperSize and PaperOrientation, you can precisely specify all aspects of the printer options.

Printer Setup Dialog

The Report Designer Component runtime has been enhanced with a new Printer Setup Dialog box. You invoke this dialog box using the new PrinterSetup method under the IReport interface.

Searching

String Search

You may now search for strings in text objects, field objects, subreports, cross-tabs and OLAP grids. You may now search field names (or object names if fields are displayed with these names instead) for strings. String searching also allows you to mark all report objects that match a string and then manipulate all these objects as a group.

Search By Formula Support

You can now search by formula in the Report Designer Component runtime. The Report Viewer will contain a boolean search button if you install the Search Expert.

Summarizing

Inserting Subtotal, Summary, Grand Total

Previously, in the RDC design time, you had to first find the database field you wanted to summarize and then right-click directly on that field before you could insert a subtotal, summary or grand total. In Version 8 you don't have to do that any more. Now you simply right-click on an empty space in the design time report and pick from a list of fields to summarize.

Additionally, when you insert a subtotal for grouped data, you now have these two options:

1. to insert a subtotal or summary field all groups, or
2. to insert a grand total field.

Note that as before, if you are creating the summary for grouped data, and in the design view you have selected an object that cannot be summarized, then the menu item Insert | Summary will be disabled. The same applies to subtotal and grand total fields.

Inserting Running Total

The RDC design time now has a shortcut for creating running total fields based on a selected database field.

You simply right click a database field and select Insert Running Total from the shortcut menu that appears. This will invoke the Create Running Total Field dialog box which automatically sets up the running total for the selected field.

Percentage Summary Fields

The Report Designer Component now supports the new Percentage Summary field. A Percentage Summary Field enables you to show a subset of data as a percentage of the entire set. For example, you can use it to show the percent contribution of each sales rep to the sales for the entire company. Formerly this had to be done using formulas.

The following changes have been made to the Report Designer Component runtime to support Percentage Summary fields:

- the type CRSTDPercentage has been added to the enumerator type CRSummaryType,
- the functions Add and Delete have been added to the SummaryFieldDefinitions class.

Viewer

Select Expert Support for the Report Viewer

The Report Viewer now supports the Select Expert when used in conjunction with the RDC runtime.

Web

Hyperlinks

You may now select an object in the RDC design time and create a hyperlink to a Web site or email address.

To create a hyperlink to a Web site or email address, you simply right-click an object (e.g., text, bitmap, chart) in the Designer and select Format from the shortcut menu. Go to the new Hyperlink tab in the Format Editor and set up your link.

For further information, please refer to the section “Linking to a Web Site or Email Address” above.

Speed Up Loading of First Page

In the RDC runtime, you can now have the program load the first page of a report even before the total page count is calculated. You can use a placeholder for the total page count for Page N of M, and when the total page count is ready, the program will insert it back to each page.

To support this feature:

- the PlaceHolderOptions property has been added to the IPageEngine interface,. You may use this property to specify whether or not you want to delay the total page count generation.
- The IsMissingTotalPageCount property has been added to the IPage interface,. You can use this property to find out whether a page is missing the total page count.

This feature is particularly useful if you use the Report Designer Component runtime to write Web reports, where the end-user can benefit from this improvement and view the first page of a large report much sooner than before.

Web Support

You may now access the Seagate Software Web page through the Report Designer Component for online registration, support, product news, and updates. To access the web page, right-click anywhere in the Designer, and choose Help | Seagate Software on the Web.

A Closer Look at Some of the New Features

The following topics are discussed in this section:

Report Variables

The RDC now includes Report Variables that are "smart" variables that can be used to maintain state when you are coding things like running totals based on a Section Format event. These variables "understand" what's going on inside the report formatting process at any given time, and they enable you to do more of your report-oriented coding exclusively in VB. The discussion that follows explains why you might need to use report variables.

The RDC uses a multiple pass model to generate its reports. The number of passes the program needs to generate a given report depends on the complexity of the report. There is no "one size fits all" answer to the question, "How many passes will it take?"

During each pass, various sections in the report get fired. A Details section might get fired several times for a single record. This might happen, for example, when you have selected the "Keep Section Together" option and the program jumps back and forth between pages as it determines on which page it should print the section.

If you use a standard VB variable to calculate running totals, that variable is unable to use the built-in intelligence of the RDC. When a section is fired multiple times, the variable gets incremented each time the Format event for that section is trapped. In this case your calculations will be distorted. When you create a running total in the RDC using a Crystal Running Total field, however, the program knows to increment the field value only once for each record, regardless of how often the section is fired.

> ***Note: The maximum allowable size for a string in a string Report Variable is 255 including the null byte. If you set a string value to more than 255, the string will be truncated.***

For further discussion on "what happens when" during the report creation process, see the Report Processing Model appendix in the *Seagate Crystal Reports User's Guide.*

Using Report Variables

Rather than using the Dim statement to declare a Report Variable, you declare it by invoking the new AddReportVariable method when the Report Initialize event fires. This method calls for you to supply a data type and variable name as parameters. For example:

```
Me.AddReportVariable crRVCurrency, "Total"
```

> ***Note: Make certain you enclose the name of the variable in double quotes.***

Once you have declared the variable, you can get its value using the GetReportVariableValue method that is part of the Report object. You can set or increment the variable using the SetReportVariableValue method. For example:

```
Me.SetReportVariableValue “Total”, Me.Field3.Value +
Me.GetReportVariableValue(“Total”)
```

Note: Please note that Report Variables are only available in RDC runtime. If you save a report as a .RPT file to use with standalone Seagate Crystal Reports, you will lose the capability of the report variable.

For further information on Report Variables, see the Report Variables sample application in the \\SeagateSoftware\ReportDesignerComponent folder. For a description of the sample application, see *Report Variables, Page 42*.

Report Creation API

The RDC now includes Report Creation API methods. These methods enable you to create ad hoc reports at runtime using nothing but code. The methods can be used within Visual Basic, Microsoft Office, or in any environment from which you can access an automation server.

The RDC also provides a Report Creation Wizard object that your application can call at runtime to enable your users to create reports at runtime using an intuitive design environment. The Wizard even attempts to identify your data source so it can make the data selection automatically.

The Report Creation API methods provide the application user with Report Designer capabilities even if they are not themselves registered users of Seagate Crystal Reports. For this reason, a new licensing model is in effect for these methods only. Please refer to the license material available from the help menu in the product for further information.

Fields that bind to data sources only at runtime

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

This functionality makes it easy to build reporting into n-tier applications where you have on-demand recordsets and non-continuous data connections. It also makes it easy to build a single report that can work with a variety of recordsets, making development and maintenance more efficient.

To take advantage of this functionality, you simply

1. Add the RDC to your project. For further information, see *Adding the RDC to your project, Page 37*.
2. Choose the Unbound Fields option from the Field Explorer in the RDC.
3. Choose a field object by data type and drag it into position where you want it to appear on your report.
4. Set the Name property of the field object to match the name of the field in your recordset that you want it to bind to.

5 Continue until you have all your fields in place.

6 Write the code to activate the binding. For our code example, we will bind to an ADO recordset.

In the code, you need to set up Object variables for both your recordset and your report. For example:

```
Dim rs As New ADOR.Recordset

Dim Report As New Crystal Report1
```

If the recordset doesn't yet exist, you must populate by passing it a selection string. For example, you could pass a string similar to the following as part of the Form Load event.

```
Rs.Open "select * from customer",  "xtreme sample database"
```

7 Finally, you need code to actually create the binding. You do that with the new Report object method AutoSetUnboundFieldSource, using code like this:

```
Report.AutoSetUnboundFieldSource crBMTName
```

That's all there is to it.

Thin-client applications and meta-layer data

The discussion of unbound fields makes it look like it is extremely easy to create unbound report templates that bind to their data sources only at runtime. It is easy, but that easiness masks, somewhat, the power of this new capability. It should not be underestimated. You can use that capability to build thin-client applications that report off of meta-layer data.

If you have been interested in having your applications utilized by larger corporations, you know that you have been forced to consider a more server or multi-tiered approach. Organizations interested in keeping up with the times have been forced to move to larger customized applications for solutions like ERP systems. This combination of technology adoption has required developers, and the tools they use, to advance beyond simple applications that connect directly to a data source.

From a reporting perspective this means that you need a way to take advantage of the meta-layers within an ERP solution.

The meta-layer

A meta-layer is an intermediate data layer that holds a subset of the data in the original data source. The data in the meta layer typically has already been summarized, manipulated, and / or cleansed. Meta-layers are used when it becomes too complicated for users to understand or connect to the raw data itself.

The application and the reports

Your other challenge is based on the customization of these systems. There is no standard database format; it is different for every company. To provide a solution, you must create client server applications that can extract the data from a meta-layer and then utilize the extracted information within the reports you provide for your users. Seagate Crystal Reports 8 takes advantage of two features to provide an answer:

- The ability to link a report to an ADO record set and
- The ability to create a report template using unbound fields.

Linking a report to an ADO recordset is not a new feature, but it ties in well with the new Unbound Fields capability. Think about it. An unbound report field:

- is a field type that you add to a blank report during the development of an application,
- has no data source until the user executes the report at runtime, and
- acts like a placeholder on the report in situations where you don't have access to the user's data source.

This is the case, every time you write an application that is meant to connect to a customized ERP type application! Now you have a solution:

- The final application extracts data from the meta-layer and writes it to an ADO object.
- The RDC connects to the ADO object at runtime and populates the Unbound Field objects.

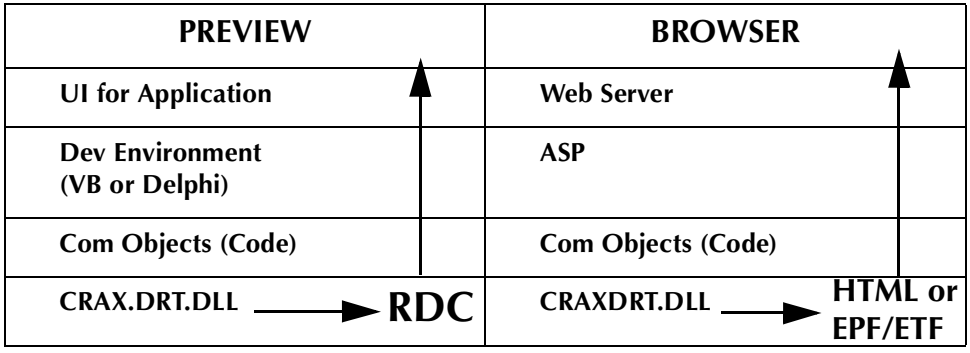
By designing your application using unbound report templates, you can still give users the ability to generate reports against this data, even though at design time the data doesn't exist.

Web-based thin-client solutions

You can also use this entire solution over the web for a true thin-client implementation You can use the same set of code (Com objects) to extract the data from the custom system and its meta-layer, and pass it to an ADO object. The CRAXDRT (using unbound fields) can then access this ADO record set and publish the report.

The main difference is that the report ends up being produced in an EPF format instead of RPT and it gets fed back to the user via the Web server in a viewer rather then in a preview window of an application.

To provide such a web solution, you would need to do your development in an ASP environment rather than a Windows-based development environment like VB, VC++ or Delphi.



Field Mapping

When you first create a report, the report draws its fields from the data source as it exists at that time. It uses the structure of the database (number of fields, field position, data type, etc.) to identify and select those fields you want placed on the report. If the structure of the database gets changed (by adding or deleting fields) after you create the report, you need to adapt the report to the new structure. The RDC now has a robust set of Field Mapping tools that you can use to accomplish this.

The FieldMapping event

In the past you have been able to use the Verify method in the Database object to verify that the structure of the data source matches that of the data source used in the report. This checks for prior changes to the structure. Now you can determine if the data source structure gets changed while you're verifying it by trapping the new FieldMapping event. The RDC fires a FieldMapping event when there is a change to the structure during the verify process.

The FieldMappingData object

The RDC also has a New FieldMappingData object that enables you to bind a report field object to a data source with a different table and / or field name, or with a different data type than that in the report. The FieldMappingData object has four read / write properties that enable you to map the changes in the data source to the database field objects in your report.

2 Integration Methods

What you will find in this chapter

OVERVIEW

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

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

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

INTEGRATION METHODS

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

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

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

Method	*Introduced in version*	*Project Reference Name*	*Description*
Report Designer Component	Seagate Crystal Reports 7	Crystal Report 8 ActiveX Designer Runtime library (craxdrt.dll)	32-bit only COM object model, dual interface, apartment model
Crystal Reports Automation Server	Seagate Crystal Reports 6	Crystal Report Print Engine 8 Object Library (cpeautl6.dll / cpeaut32.dll)	32- and 16-bit COM object model, dispatch only Interface for C developers
Crystal Report Print Engine API	Seagate Crystal Reports 5	Crystal Report API Interface, declares encapsulated in GLOBAL32.bas	
Crystal ActiveX Control (OCX)	Seagate Crystal Reports 4.5	ActiveX Control	32- and 16-bit OCX

USING THE BEST TECHNOLOGY – THE REPORT DESIGNER COMPONENT

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

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

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

The RDC consists of three main components:

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

Visual Basic and Office 2000 (VBA) Developers

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

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

Visual InterDev Developers

Many Visual InterDev developers are experienced using Visual Basic and are therefore familiar with integrating reports into database applications. Developers can use the same RDC automation server and Report Viewer in Visual InterDev as they use in other COM-based environments such as Visual Basic and

Visual C++ to access the features of Seagate Crystal Reports. This provides one object model that offers the same control over the report engine in web applications as well as Windows applications. Developers can save time since they can build a report once and move it to either the desktop or the Web. Report Integration controls then give Visual InterDev developers an easy way to integrate reports into ASP pages.

Visual C++ and other COM-based Developers

If you use development environments that support COM-like Visual C++ and Lotus Domino can:

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

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

Evolution of Integration Capabilities of Seagate Crystal Reports

To understand the advantages of using the RDC, it's also important to be aware of the technology from which it evolved.

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

OCX

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

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

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

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

Crystal Report Print Engine API

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

Using these direct Report Engine calls, you could access more functionality than offered with the OCX. However, because it was designed for C/C++,

- it wasn't easy to use the API inside Visual Basic and
- some of the API wasn't available because of Visual Basic limitations in creating C style structures.

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

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

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

Crystal Report Automation Server

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

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

When it was released, it too was the best technology available in its time.

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

No new features have been added to the Crystal Report Automation Server since it was introduced in Seagate Crystal Reports 6. We've ensured backward compatibility in Seagate Crystal Reports 8 to allow you to leverage applications created in the previous version. However, the Crystal Report Automation Server no longer exposes all of the events and properties of the Report Engine in Seagate Crystal Reports 8. Because the Report Designer Component is COM-based, and an evolution of the Crystal Reports Automation Server, you will find it easy to migrate from the Crystal Reports Automation Server to the RDC.

Report Designer Component (RDC)

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

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

THE BEST TOOL FOR YOUR NEEDS

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

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

Visit http://community.seagatesoftware.com to find the resource that best suits your needs. Chapter 9, *Migrating to the RDC from the OCX, Page 117* of this guide shows you how to move quickly to the RDC from the OCX.

The RDC is the premium development method, and it will continue to be enhanced for developers. The latest version of the RDC is included in Seagate Crystal Reports 8.

3 Introducing the Report Designer Component

What you will find in this chapter

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. You can open, design and customize reports within the Visual Basic IDE. Additionally:

- You can open, design and customize reports within the Visula 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, you have complete control over your report designs, using familiar Visual Basic code.
- Report distribution is simplified through a small component count and free runtime.
- Reports can thus be packaged within the applications' executable or stored outside the application in the traditional (.rpt) format.

Design time

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

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

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

Runtime

The user interface provided by the Report Designer Component at design time does not appear in your application at runtime, or when it is compiled into an executable file. Instead, the Report Designer Component is accessed directly by your Visual Basic code. The Report Designer object model provides a complete object hierarchy for direct manipulation in Visual Basic.

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

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

RDC Architecture

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

Report Designer

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

Automation Server

The Automation Server (craxdrt.dll)—known as the RDC runtime engine—is an extensive object model with hundreds of properties and methods for you to use to manipulate the report at runtime.

Crystal Report Viewer

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

Component	***Description***
Crystal Report Designer UI Component	This is a COM (Component Object Model) component that provides the user interface at design time for the user to interact with and create or modify the report.
Crystal Report Designer Design Time Component	This is an underlying COM component that provides services for the user interface component.
Crystal Report Designer Run Time Component	This is the component that encapsulates all of the report objects and is responsible for doing all of the data processing and report layout.
Active Data Driver	This is a data access driver that provides access to various types of object data sources including DAO, RDO, and ADO.
Crystal Report Viewer for ActiveX	This component is an Active X control which can be drawn on a form and manipulated at design time. It provides a rich object model which can be used to modify user interaction with the report at runtime. This component is required only if a developer wants to provide on-screen display of reports at runtime.

Component	*Description*
Data Set	One of the following: • Data Access Object (DAO) Recordset • Remote Data Object (RDO) Resultset • Active Data Object (ADO) Recordset • VB Data Environment • Crystal Data Object (CDO) • Crystal Data Source Type Library object • ODBC Direct These objects do not need to be valid at design time, for example you could construct a report template at design time without the data being available. This is handled through *Data Definition Files, Page 62.* However, the data set objects must be present and valid at runtime to generate a report.
VB Form	The Seagate Crystal Reports Report Viewer Control must be embedded on a Visual Basic Form in order to display the report on screen. The Create Report Expert can automatically add a Form with the Report Viewer embedded to the project when you finish designing a report with the Expert.

These objects do not need to be valid at design time, for example you could construct a report template at design time without the data being available. This is handled through *Data Definition Files, Page 62*. However, the data set objects must be present and valid at runtime to generate a report.

The Seagate Crystal Reports Report Viewer Control must be embedded on a Visual Basic Form in order to display the report on screen. The Create Report Expert can automatically add a Form with the Report Viewer embedded to the project when you finish designing a report with the Expert.

Which components to use - Possible Development Scenarios

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

Using the Report Designer with the Automation Server	VB and VBA only. Want to design reports within IDE and manipulate the reports with code. No need to preview the reports. Reports can be printed or exported. Can use Crystal Formula Language, VB or VBA code for expression in report.
Using all three RDC components	VB and VBA only. Want to design reports within IDE and manipulate the reports with code. Reports can be previewed, printed or exported. Can use Crystal Formula Language, VB or VBA code for expression in report.

Using the Automation Server and the Crystal Report Viewer	VB and VBA, C++, Visual InterDev, Visual J++. Want to design reports using Crystal Reports, preview from application, and manipulate the reports with code. Reports can be printed or exported. Must use Crystal Formula Language, VB or VBA code for expression in report.
Using the Automation Server by itself.	VB and VBA, C++, Visual InterDev, Visual J++. Want to design reports using Crystal Reports and manipulate the reports with code. Reports can be printed or exported. Must use Crystal Formula Language, VB or VBA code for expression in report.

Adding the RDC to your project

The installation program will attempt to add the menu item ADD CRYSTAL REPORTS 8 to the Project menu of the VB IDE when you install the RDC. If that menu item appears, you simply select it to add the RDC to your project.

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

1 Choose Components from the Project menu.

2 Click the Designers tab in the Components dialog box when it appears.

3 Select Crystal Reports 8 on the Designers tab. This will add the menu item to the Project menu.

4 Finally, select Add Crystal Reports 8 from the Project menu.

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

HOW TO GET ADDITIONAL INFORMATION

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

Seagate Crystal Reports User's Guide, Page 38

Help System, Page 38

Community on the Seagate Software Web Site, Page 38

Object Browser, Page 39

Properties Window, Page 39

Sample Reports, Page 39

Sample Applications, Page 39

Report Wizard, Page 42

Complex Applications, Page 44

Seagate Crystal Reports User's Guide

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

Help System

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

Community on the Seagate Software Web Site

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

The web site offers these support tools and more:

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

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

Object Browser

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

Properties Window

The RDC exposes each object in an RDC report. This means that you can inspect all of the design time properties and options in the Properties Window of the VB IDE. Simply select an object in the report (just like you'd select any command button or text box on a form) and you can see and set its various properties visually.

Sample Reports

The program ships with dozens of sample reports that you can use to learn sophisticated reporting techniques. If you did the full installation, you'll find these samples in the \Crystal Reports\Samples\Reports folder.

Sample Applications

We've created this special section on Sample Applications because we want to call them to your attention!

Seagate Crystal Reports ships with over 18 Visual Basic sample applications that show you how to use the RDC in real-world situations. The applications are well-commented so you can easily follow what is going on with the code, even if you've never used the RDC before.

If you did a complete install and used the default installation settings, you will find the sample applications in the Program Files\Seagate Software\Report Designer Component\Code\Visual Basic folder. There are samples for C++, Delphi, and the Web in the \Code folder as well. The sample applications provide you with one of the most direct and tested methods for learning the RDC.

These are not "last minute" applications, prepared as an afterthought minutes before shipment; they are well-planned, carefully-constructed applications designed specifically to showcase challenges you face as a developer.

The applications contain a lot of usable code that can be modified with little effort and plugged into your own programs. They also present a lot of great ideas for using the RDC. By studying these ideas, you may be able to generate new business by offering clients sophisticated capabilities they didn't know were available.

Visual Basic Samples

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

Simple Application

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

ADO Connection Methods

This application demonstrates the two new methods that provide convenient ways to add ADO data sources to a report:

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

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

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

Change Runtime Location of OLE Object

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

This application takes three different OLE objects;

- a bitmap,
- part of an Excel spreadsheet, and
- part of a Word document;

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

Employee Profiles

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

Inventory Demo

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

Load Picture Demo

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

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

- Using the Crystal Formula Language
- Using VB code in the Section_Format event procedure

This example shows how to do the calculations in the Section_Format event using VB.

Microsoft Office Samples

These samples can be found on the Seagate Software website at http://community.seagatesoftware.com.

No Data in Report Event

This application demonstrates the new NoDataInReport event.

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

Printer Settings

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

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

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

Report Object Creation API

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

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

- You can add these report objects at runtime to an existing report that you created in the RDC.
- You can also create a blank report at runtime and add these objects to that blank report. This is the methodology shown in this sample.

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

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

Report Variables

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

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

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

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

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

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

Report Wizard

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

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

Search and Select Experts

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

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

- Two command buttons on the main form display a search or select wizard with predefined fields and ranges already set for the user. The only displayed fields and ranges are those that have been added at runtime.
- The Search and Select Expert buttons on the top right part of the viewer are over-ridden, allowing you to specify default criteria to search for. Alternatively, the default search and select experts can also be displayed.

Simple Demo

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

Unbound fields

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

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

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

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

Viewer

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

Viewer Runtime options

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

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

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

Complex Applications

The following three applications are large and complex. They each demonstrate many of the RDC capabilities, often in new and exciting ways.

Xtreme Mountain Bikes

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

Pro Athlete Salaries

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

If you will note, the Designer does not appear in the Project Explorer. This is because:

- the report is actually a template report created at runtime using Unbound Fields and the Create API, and
- the fields are bound at runtime to a custom recordset generated at runtime based on user input.

None of this report design is done visually—it is all done through code.

First Class Hotels

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

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

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

The Room form, frmRoom, has these features, among others:

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

The Price form, frmPrice, has these features and others as well:

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

This application uses so many of the methods from earlier releases and this release as well (* = new in version 8), that we've listed them to help you determine if this application has the answers you're looking for:

Crystal Reports Viewer Object

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

Crystal Reports Report Object

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

4 Quick Start for using the RDC

What you will find in this chapter

OVERVIEW

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

> ***Note: For information on using the RDC with other development environments, See Chapter 10, Working with Visual C++, Visual InterDev and Lotus Domino, Page 125.***

The following three sections were designed to get you up and running quickly with the RDC:

Bullet Point Quick Start (page 48)
If you only need minimal instructions for using the RDC, this section is for you. It's a high-level, no-frills bullet-point explanation of the steps you need to take to create, view, and manipulate a report through the RDC.

Open an Existing Report (page 50)
If you have created a report in the stand-alone version of Seagate Crystal Reports and you want to open it through the RDC and preview it in the Report Viewer, you'll find this section useful. It's a lower level, step-by-step approach.

Create and Modify a Report (page 51)
If you want to create, preview, and manipulate a report through the RDC, you'll find this section extremely useful. This is also a lower-level, step-by-step approach. It mirrors the procedure of the Bullet Point Quick Start and "fleshes out" each of the bullet points.

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

BULLET POINT QUICK START

Adding reporting capabilities to your application using the RDC is extremely easy if you follow these steps:

- Add the RDC to your project
  - — If you do a complete install, the option Add Crystal Reports 8 should appear on the Project menu in the VB IDE. If it does, select it. That adds the RDC.
  - — If it doesn't, select Components from the Project menu and check the Crystal Reports 8 option on the Designers tab. Then select Add Crystal Reports 8 when it appears on the Project menu.
- Select a data source

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

- Create a report

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

- Add the Report Viewer to your project

  When you select Finish in the Standard Report Expert, the program automatically adds a form to your project and installs the Report Viewer on the form. It also writes the necessary code for displaying your report in the viewer and resizing the viewer whenever the form is resized.

  — If you don’t want the Viewer installed automatically or if you don’t want the Viewer form to be your start-up form, you can change those actions in the Default Settings dialog box.

  — You get to that Dialog by right-clicking in the RDC and selecting Designer and then Default Settings from the shortcut menu when it appears.

- Create a UI for users to modify the report at runtime

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

- Write the code that does the modifications

  Now you can write the code that ties the users UI selections to the report.

  — Each object in the report is exposed in the Properties window of the VB IDE.

  — Double-click any object and you can write code for the object.

  — Double-click on a section of the report and you can write code that affects the section and the objects in the section.

  — By referencing an object name in your UI code or the object index through one of the many Collections, you can manipulate that object directly based on user input.

- Preview, print, or export the report

  When you’re all finished, run the application and preview, print, or export the report using the UI controls you’ve set up to do it.

That’s all there is to it. Now you have the most powerful reporting capabilities in the world running as a piece of your application.

OPEN AN EXISTING REPORT

In this example, you will open an existing report file in Visual Basic. This will demonstrate how to import existing reports (*.RPT files) and allow you to become familiar with the Report Designer Component user interface. This tutorial uses Seagate Crystal Reports 7 and Microsoft Visual Basic 6.0.

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

Open a Report directly within Visual Basic 5.0 or 6.0

1. Open Visual Basic and create a new Standard EXE project by selecting Standard EXE from the start up dialog or selecting it from New Project under the File menu.
2. Add the Report Designer Component to Visual Basic if not already added during the installation process.
3. From the Project menu, select Components…
4. Click the Designers tab and check Crystal Reports 8. Click Apply and then click Close. The Report Designer is now available in this project and any projects you create in the future.
5. Now we need to insert the Report Designer into the project form. On the Project menu, point to More ActiveX Designers and then click Add Crystal Reports 8.
6. The Seagate Crystal Report Gallery appears displaying the different types of Report Experts that are available. Since you will be opening an existing report file, click From an Existing Report. Click OK.
7. Browse to the report called World Sales Report at \Crystal Reports\Sample\Reports\General Business\. Click Open. Depending on your setup, you may be presented with a dialog that asks you about adding a form at runtime. Click OK for now.
8. The Report Designer Component is added to your project and, in a few seconds, the report layout is displayed in the design window. Report files created in any version of Seagate Crystal Reports can be imported in this manner. Before you run the report, right-click on Form2, and select View Code. You should see Visual Basic code that looks like this:

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

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

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

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

CREATE AND MODIFY A REPORT

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

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

Add the RDC to your project

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

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

3 From the Project menu, select Components…

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

5 Now we need to insert the Report Designer into the project form. On the Project menu, point to More ActiveX Designers and then click Add Crystal Reports 8. (In some environments, Add Crystal Reports 8 will appear as an option itself on the Project menu. In such a case, choose that option.)

Select a Data source

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

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

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

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

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

Create a report

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

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

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

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

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

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

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

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

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

Add the Report Viewer to your project

1 The Report Expert presents you with the option of adding a form with the Crystal Report Viewer control and setting this as the start-up object. Click OK to accept the defaults. The Expert will format the report and display it in the design window.

2 Click the Start button on the Visual Basic toolbar or press F5 to run your project. After a few seconds, you will see a form displaying the finished report in the Crystal Report Viewer.

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

Create a UI for users to modify the report at runtime

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

Write the code that does the modifications

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

1. Close the Crystal Report Viewer.
2. From the Project menu select References...
3. Check the Microsoft ActiveX Data Objects Recordset 2.0 Library and click OK. You shouldn't need to do this again in future projects.
4. Double-click Form2 from your Project Explorer to open it. This form contains the Report Viewer. Double-click on it to see the code in the main window. It should look something like this:

This code should appear in the General Declarations section

```
Dim Report As New CrystalReport1
```

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

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

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

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

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

```
Dim Report As New CrystalReport1

Dim rs As New ADOR.Recordset

Private Sub Form_Load()

     Screen.MousePointer = vbHourGlass

     rs.Open "Select * from customer where country = 'USA'", "xtreme sample
database"

     Report.Database.SetDataSource rs

     CRViewer1.ReportSource = Report

     CRViewer1.ViewReport

     Screen.MousePointer = vbDefault

End Sub
```

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

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

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

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

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

```
Private Sub Section5_Format(ByVal pFormattingInfo As Object)

    Text5.SetText "Here is some text from my app"

    Text5.Font.Italic = True

    Text5.Font.Bold = True

    Text5.Font.Size = 14

End Sub
```

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

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

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

```
Private Sub CRViewer1_DrillOnGroup(GroupNameList As Variant, ByVal DrillType
As CRVIEWERLibCtl.CRDrillType, UseDefault As Boolean)

MsgBox "You're drilling down on the " & GroupNameList(0) & " group!"

End Sub

Private Sub CRViewer1_PrintButtonClicked(UseDefault As Boolean)

                MsgBox "You clicked the Print button!"

End Sub
```

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

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

5 Data Access

What you will find in this chapter

OVERVIEW

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

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

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

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

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

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

THE DATA EXPLORER

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

ODBC

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

Note: To use these drivers, you must have installed the full Seagate Crystal Reports product.

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

You will need to supply:

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

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

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

To connect to an existing ODBC data source, you simply:

1 Choose the data source.

2 Choose the table(s) that contain the data you want to use.

3 Link the tables (if required) using the Visual Linking Expert.

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

At this point you simply drag the fields you want onto your report and position them where you want them.

File Directory

This node lists standard PC data sources that reside locally. If your database isn't listed, you can browse for your file via the FIND DATABASE FILE option.

More Data Sources

The More Data Sources node is where all the action is. You use this node to select data sources that can be accessed through OLE DB and native drivers. This includes:

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

ACTIVE DATA SOURCES

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

ADO

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

ADO can be used to connect to any ODBC or OLE DB compliant data source when you design your report in the Report Designer Component. The resulting ADO Recordset is not directly available from the Report Designer Component, but it can be replaced by alternative Recordset objects. For example, if your Visual Basic application allows users to make choices that can affect the set of data displayed by a report, simply create a new ADO Recordset at runtime, then pass it to the Report Designer Component using the SetDataSource method. The only restriction is that the fields in the new Recordset match the fields originally used to design the report. For more information on using the SetDataSource method, see *SetDataSource method, Page 64*.

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

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

DAO

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

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

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

RDO

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

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

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

Crystal Data Object (CDO)

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

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

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

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

Crystal Data Source Type Library

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

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

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

CDO vs. the Crystal Data Source Type Library

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

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

Data Definition Files

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

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

While data definition files provide great flexibility in your reporting, and they are still supported in Version 8, they have been superseded with a more elegant technology, Unbound Fields.

DATA ENVIRONMENTS

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

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

Microsoft Data Link (UDL) Files

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

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

To create a UDL file:

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

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

Report Templates using Unbound Fields

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

This technology replaces the need for data definition files. Data definition files are still supported in the Report Designer Component however. For additional information on using Unbound Fields, see For additional information on using Unbound Fields, see *Fields that bind to data sources only at runtime, Page 23* or the *Unbound fields, Page 43* or *Pro Athlete Salaries, Page 44* sample applications found in the \Report Designer Component\Code\Visual Basic folder.

DATABASE DRIVERS

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

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

Crystal Active Data Driver

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

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

Passing the Recordset

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

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

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

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

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

SetDataSource method

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

When RDC uses the Active Data Driver

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

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

Determining if the RDC is using the Crystal Active Data Driver

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

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

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

6 Understanding the RDC Object Model

What you will find in this chapter

OVERVIEW

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

A high level overview

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

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

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

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

- Manipulating the section means doing things like setting it's background color or its height.
- Manipulating an object means doing things like changing a font color in a field object based on a condition being true or changing the report title in response to user input.

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

On this very basic level, that's all there is to it.

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

That's all you need to know to get started.

The next level

A report can have more than five sections. Many more.

- You can create these sections yourself if you need additional sections.
- The RDC will create some for you whenever you group or summarize data.

Sections are numbered consecutively as they are created.

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

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

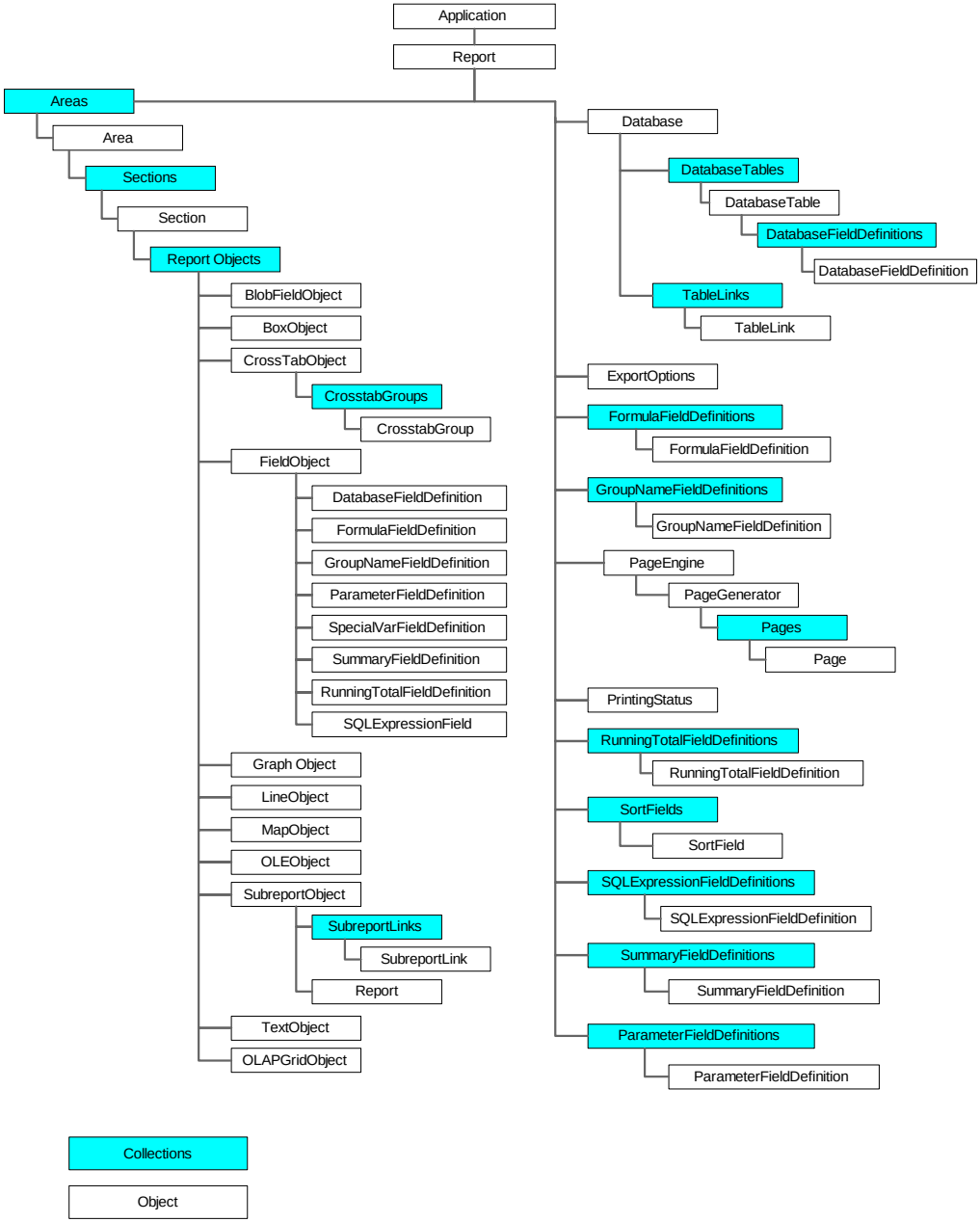
You can place a variety of Report Objects in the sections of your report.

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

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

THE PRIMARY OBJECTS AND COLLECTIONS

The following objects and collections are discussed in the section:


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

The Application object

The top of the object model is the Application object. Since it's at the top, the RDC always expects it to be there, so there's no need to reference it in your code (generally). There are some specific instances where you must reference it. For more information, see *The Application Object, Page 73*.

The Report object

Below Application is Report. Report contains a lot of Collections and a lot of objects. For more information, see *The Report Object, Page 74*.

The Areas collection

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

The Sections collection

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

What might cause that need?

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

Thus, unless you have a special need, you can think of Sections as having Report for a parent. If you have the need, or if you want to be very specific, you can also think of Sections having Area for a parent. However you think of this part of the object model is fine. As long as you have five sections and five areas, though, you can go through Areas to get to Sections or you can go directly to Sections. Your code will work either way. For more information, see *The Sections Collection, Page 78*.

The ReportObjects collection

Now that we have Areas and Sections clarified, we need to consider ReportObjects.

ReportObjects is a collection that can contain a lot of things. With the exception of Areas and Sections, everything else in a report is a report object: Database field objects, Parameter field objects, Text objects – they're all included. You'll note that we used a small "r" and a small "o" for our spelling of "report object". That's because there is no "ReportObject" in the ReportObjects collection. The ReportObjects collection is just a convenient way to think about these objects and it can be useful in your code.

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

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

You can access an object through the ReportObjects collection or through its namesake collection. This provides you much flexibility in your coding. See *The ReportObjects Collection, Page 76*.

The FieldDefinitions collection

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

Following this all the way back up the hierarchy, then, we can see that a ParameterFieldDefinition:

- Is a member of its namesake collection, ParameterFieldDefinitions.
- It is also a member of the FieldDefinitions collection.
- It is a member of the ReportObjects collection as well.

If this seems like a lot of collections to contend with for a single object, be assured that there's a very good reason for this structure.

The Subreport Object

One very special member of the ReportObjects collection is SubreportObject. A subreport is a report within a report. Thus it is, itself, a complete report with its own Areas, Sections, and ReportObjects. If that sounds confusing, it really isn't. Once you access a Subreport object, you deal with the objects in the subreport just as you deal with objects in the main report. You just reference them a little differently. For more information, see *The Subreport Object, Page 75*.

So far, we've taken a look at the object model from the report side of things. We haven't been involved with the data that actually gives value to a report.

The Database Object

The Database object represents a data source for the report. But a data source doesn't mean all the data available. It means only a subset of data that is useful for the report. That subset can be defined as collections of tables and, within each of those tables, collections of fields. When you think about a data source in this way the data side of the RDC object model becomes clear. For more information, see *The Database and Database Tables Objects, Page 74*.

The DatabaseTables collection

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

The DatabaseFieldDefinitions collection

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

SPECIAL CONSIDERATIONS

Object considerations

A number of objects require a little further discussion.

The Application Object

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

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

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

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

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

Note: The sample call to CreateObject above uses a version independent Prog Id for the Report Designer Component. The correct Prog Id for this version of the Report Designer Component is CrystalRuntime.Application.8, but the version independent Prog Id should use the most recent version of the component installed on your system.

The Report Object

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

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

Example:

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

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

The Field Object

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

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

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

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

For examples of how to use either of these methods, see *Referencing objects in a report, Page 84*.

The Database and Database Tables Objects

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

The Database object is available directly from the Report object and represents the data source used by the report. This data source can be changed using the Database object's SetDataSource method. In addition, the Database object provides the Tables property, a read-only property that gets the DatabaseTables collection of DatabaseTable objects. You have access to:

- log on information (for password protected systems),
- database driver names, and
- database locations.

through a DatabaseTable object. Use code similar to the following to work with the Database and DatabaseTable objects:

```
Dim Report As New CrystalReport1
Dim tableName As String

Report.Database.Verify() ‘ Verifies a valid connection to the database

For Each dbTable In Report.Database.Tables

   tableName = dbTable.Name

Next
```

The Subreport Object

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

Note: You cannot print or export a subreport, or preview it in the Report Viewer outside of its primary report.

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

```
Dim Report As New CrystalReport1
Dim subReport As SubreportObject

For Each sect In Report.Sections

    For Each rptObject In sect.ReportObjects

           If rptObject.Kind = crSubreportObject Then

                  Set subReport = rptObject
                  subReport.BackColor = RGB(255, 0, 255)
                  Set subReport = Nothing

           End If

    Next

Next
```

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

The CrossTabObject

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

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

Once you have a CrossTabObject, you can manipulate it and the report objects it contains much like you would when working with any other report. Because of the complexity of a CrossTabObject and its specialized handling of data, however, fewer properties and methods are exposed than you find with a Report itself. Please consult the Object Browser or the help system for further information on CrossTabObject members.

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

```
Dim Report As New CrystalReport1
Dim xtObject As CrossTabObject

For Each sect In Report.Sections

                For Each rptObject In sect.ReportObjects

                      If rptObject.Kind = crCrossTabObject Then

                            Set xtObject = rptObject
                      xtObject.BorderColor = RGB(255, 0, 0)
                            xtObject.HasDropShadow = True
                            Set xtObject = Nothing

                      End If

                Next

Next
```

Collection considerations

A number of collections require further discussion.

The ReportObjects Collection

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

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

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

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

```
Dim Report As New CrystalReport1
Dim sect As Section
    Dim rptObject As Object
    Dim objKind As String
    Dim lastSectWithObject As Integer

lastSectWithObject = Report.Sections.Count
Set sect = Report.Sections.Item(lastSectWithObject)

Do While sect.ReportObjects.Count <= 0

                lastSectWithObject = lastSectWithObject - 1
    Set sect = Report.Sections.Item(lastSectWithObject)

Loop

Set rptObject = sect.ReportObjects.Item(sect.ReportObjects.Count)

Select Case rptObject.Kind

                Case crBlobFieldObject
                objKind = "BLOB field object"

                Case crBoxObject
                objKind = "Box object"

                Case crCrossTabObject
                objKind = "CrossTab object"

                Case crFieldObject
                objKind = "Field object"

                Case crGraphObject
                objKind = "Graph object"

                Case crLineObject
                objKind = "Line object"

                Case crOleObject
                objKind = "OLE object"

                Case crSubreportObject
                objKind = "Subreport object"

                Case crTextObject
                objKind = "Text object"

                Case Else
                objKind = "Unknown object"
End Select
```

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

The Sections Collection

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

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

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

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

For information on how to obtain and work with other objects within a Section object, such as fields and text objects, refer to *The ReportObjects collection, Page 71*.

The "Section" collection

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

Event considerations

The RDC directly supports four events:

- AfterFormatPage
- BeforeFormatPage
- FieldMapping
- NoData

These are all new events and they are discussed in detail in the Developer's online help.

The RDC supports the Format event for the Section object as well.

The Format event for the Section object

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

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

1. In the Visual Basic Project window, select the CrystalReport1 designer from the Designers folder.
2. Click the VIEW CODE button in the toolbar at the top of the Project window. A Code window will appear for the CrystalReport1 designer component.
3. In the object drop-down list box at the top left of the Code window, select a Section object that you want to apply code to. Notice that Section objects are numbered in the order that they appear on your report, from the top of the Report Design window to the bottom. For instance, if you select the Section1 object, your code will apply to the Report Header section of your report. These Section objects are labeled for you at the top of each section in the Report Designer window.

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

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

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

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

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

Note: If you are using the Format event to calculate values and you need to carry that value across sections (as in a running total, for example), you will need to use a Report Variable to store your ongoing total. Report Variables are new to version 8. For further information on Report Variables, please see *Report Variables, Page 9****.***

The Visual Basic Initialize and Terminate events

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

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

```
Dim Report As New CrystalReport1
```

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

```
Set Report = Nothing
```

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

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

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

7 RDC Programming

What you will find in this chapter

OVERVIEW

This chapter provides step-by-step solutions to a number of common runtime challenges. Most of these challenges involve:

- accessing an object, then
- changing the properties.

Rather than repeating similar code for each example, this chapter begins by presenting the two general methods you can use to access an object in a report or subreport.

The examples will then:

- Assume that you have already used one of these methods to locate the object of interest.
- Concentrate on the special considerations you may have with each of the various procedures.

SPECIAL CONSIDERATIONS

Object naming considerations

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

```
CRAXDRT.Database
```

for the object from the RDC runtime Object Library, or

```
DAO.Database
```

for the object from the DAO Library.

Scope considerations

When you specify a data source using code, it is important that you place the code in the appropriate location in your project. You can face “no data” or runtme error situations if you don’t because the report may be looking for data that has gone out of scope.

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

Typical out-of-scope problems include:

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

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

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

```
Dim Report As New CrystalReport1

Private Sub Form1_Load()
   Call SetData
       CRViewer1.ReportSource = rpt
       CRViewer1.ViewReport
    End Sub

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

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

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

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

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

Index considerations

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

Dual Interface considerations

The Report Design Component provides dual interfaces. Because of this,

- Development environments such as Visual Basic and Visual C++ can bind with vTable interfaces at compile time.
- Less robust environments (VBA, ASP, for example), can use the IDispatch interface for runtime-only binding.

It is important that you use the more efficient vTable interface if your development environment supports it:

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

When you're working with Visual Basic, you will always use the more efficient vTable interface as long as you Dim your object variables using specific class names instead of using the more generic "Object" type.

In the sections that follow (*Referencing objects in a report, below*, and *Referencing objects in a subreport, Page 87*):

- The methods shown for referencing objects explicitly provide early binding.
- The methods shown for referencing objects implicitly provide for late binding.

Two Methods to Access an Object in a Report or Subreport

Referencing objects in a report

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

- Referencing the object Explicitly is the easiest and quickest method for referencing a report object. It provides early (compile time) binding. When you use this method, however, your code is bound to the report.
- Referencing a report object Implicitly is a more generic approach. It results in late (runtime) binding. This approach, however allows you to reuse the code for other reports.
- You can get at objects in a report either explicitly or implicitly.

Explicit reference – an object in the report

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

```
Dim myReport As CRAXDRT.Report
Set myReport = New CrystalReport1
Dim myReportField As FieldObject
Set myReportField = myReport.OrderID
```

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

```
End Sub
```

Implicit reference – cycling through the sections

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

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

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

Start by getting the sections from the Main report.

```
Set CRXSections = Report.Sections
```

You begin cycling through each section in the main report.

```
For Each CRXSection In CRXSections
```

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

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

You cycle through the objects.

```
For Each ReportObject In CRXReportObjects
```

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

```
If ReportObject.Kind = crSubreportObject Then
```

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

```
Set CRXSubreportObj = ReportObject
```

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

Implicit reference – through a collection

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

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

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

You can put this code in the General Declarations.

```
Dim CrxReport As New CrystalReport1
Dim Crxformulafields As FormulaFieldDefinitions
Dim Crxformulafield As CRAXDRT.FormulaFieldDefinition
```

You can put this code in the Form_Load event procedure.

```
Private Sub Form_Load()
```

This code gets the collection.

```
Set Crxrformulafields = Report.formulafields
'You can put your code here to cycle through the collection to get the
specific object you want
```

Finally you can view the report.

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

Referencing objects in a subreport

Before you work with a subreport, you have to reference it (open it).

You can reference a subreport object in a report either explicitly or implicitly.

- Referencing the subreport Explicitly is the easiest and quickest method for referencing a subreport object. When you use this method, however, your code bound to the report.
- Referencing a subreport object Implicitly is a more generic more generic approach. This approach, however allows you to reuse the code for other reports.

Explicit reference – the subreport itself

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

```
Dim myReport As CRAXDRT.Report
Dim mySubReport As SubreportObject
Set myReport = New CrystalReport1
Set mySubReport = myReport.Orders
```

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

Explicit reference – an object in the subreport

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

```
Dim myReport As CRAXDRT.Report
Dim mySubReport As SubreportObject
Set myReport = New CrystalReport1
Set mySubReport = myReport.Subreport1
Dim subReportField As FieldObject
Set subReportField = myReport.Orders_OrderID
```

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

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

- the name of the subreport,
- an underscore, and then
- the name of the field.

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

Implicit reference

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

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

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

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

You start by getting the sections from the Main report.

```
Set CRXSections = Report.Sections
```

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

```
For Each CRXSection In CRXSections
```

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

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

You cycle through the objects.

```
For Each ReportObject In CRXReportObjects
```

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

```
If ReportObject.Kind = crSubreportObject Then
```

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

```
Set CRXSubreportObj = ReportObject
```

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

```
Set CRXSubreport = CRXSubObj.OpenSubreport
```

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

```
End If
Next ReportObject
Next CRXSection
```

RUNTIME EXAMPLES

The following topics are discussed in this section:

Working with Cross-Tabs

Modifying a cross-tab at runtime

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

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

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

```
Dim Report As New CrystalReport1
Dim xtObject As CrossTabObject

For Each sect In Report.Sections

                        For Each rptObject In sect.ReportObjects

                              If rptObject.Kind = crCrossTabObject Then

                                    Set xtObject = rptObject
                              xtObject.BorderColor = RGB(255, 0, 0)
                                    xtObject.HasDropShadow = True
                                    Set xtObject = Nothing

                              End If

                        Next

Next
```

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

Working with Data sources

Changing an ADO data source location – new methods

There are two methods new to Seagate Crystal Reports 8 that provide a convenient way to add ADO data sources to a report:

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

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

```
Option Explicit

Dim m_Report As New CrystalReport1
```

The ADO connection to the local database.

```
Dim cnn1 As ADODB.Connection

Dim datcmd1 As ADODB.Command
```

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

```
Private Sub cmdADO_Click()

    Dim strCnn As String
```

Open the data connection.

```
    Set cnn1 = New ADODB.Connection

    strCnn = "Provider=MSDASQL;Persist Security Info=False;Data Source=Xtreme
Sample Database;Mode=Read"

    cnn1.Open strCnn
```

Create a new instance of an ADO command object.

```
    Set datcmd1 = New ADODB.Command

    Set datcmd1.ActiveConnection = cnn1

    datcmd1.CommandText = "Customer"

    datcmd1.CommandType = adCmdTable
```

Add the datasource to the report.

```
    m_Report.Database.AddADOCommand cnn1, datcmd1

End Sub
```

Adding a data source and a field using AddOLEDBSource

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

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

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

```
M Report.Database.AddOLEDBSource DataEnvironment1.Connection1,"Customer"
```

Setting the data source for a subreport

This section describes how to change the database location for different types of datasources (Native, ODBC, Active Data).

You change the database location (change the data source) for a subreport in much the same way as you set the database location for the main report. Before you can change the database location for a subreport, you first need to open the subreport. See *Referencing objects in a subreport, Page 87* for information on how to do that.

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

Native connection to PC database

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

1. Get the Tables collection for the subreport.

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

2. Get the first table from the Tables collection.

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

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

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

Native connection to SQL database

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

1. Get the Tables collection for the subreport.

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

2. Get the first table from the Tables collection.

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

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

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

ODBC connection to a PC or SQL database

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

1. Get the Tables collection for the subreport.

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

2. Get the first table from the Tables collection.

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

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

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

Active data connection to a PC or SQL database

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

1 Get the Tables collection for the subreport.

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

2 Get the first table from the Tables collection.

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

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

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

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

Connecting to OLEDB providers

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

Microsoft Jet 3.51 OLE DB Provider

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

Note:

- ***At this time, it is not possible to use secured Access databases in the Crystal Report expert.***
- ***This OLE DB provider appears to only work with 32-bit Access databases.***
- ***Notice that the database table and query names do not appear in the drop-down boxes. This has been tracked. The option is to use the SQL textbox instead.***

Microsoft OLEDB Provider for SQL Server

Syntax:

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

Example:

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

Microsoft OLEDB Provider for ODBC Drivers

Syntax:

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

Examples:

For secured Data Sources:

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

For non-secured data sources type the DSN:

```
Xtreme Sample Database
```

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

Microsoft OLEDB Provider for Oracle

Syntax:

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

Example:

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

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

Connecting to a secure Access session

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

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

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

```
Dim Report As New CrystalReport1
For Each dbTable In Report.Database.Tables
   dbTable.SetSessionInfo “user”, “password”
Next
```

A word about working with secure data in reports

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

Working with Formatting

Using Report Variables

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

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

There are three methods you use with report variables.

AddReportVariable

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

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

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

GetReportVariableValue

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

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

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

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

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

Formatting a section or report object conditionally

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

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

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

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

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

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

Working with Formulas/Selection Formulas

Passing a Selection Formula at Runtime

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

- The first shows you how you can hardcode the selection formula.
- The second shows you how to use a variable.

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

Hardcoding

If you're hardcoding the selection formula, you can copy your code from your report and paste it into the code:

```
Sub Command1_Click()
CrRpt.RecordSelectionFormula = "{Customer.Customer ID} = 1"
CrRpt.Preview
End Sub
```

Using a variable

If, however, you would like to replace the value with a variable, the code would look like the following:

```
 Sub Command1_Click()
'x can be a textbox being populated with an integer
Dim x as integer
CrRpt.RecordSelectionFormula = "{Customer.Customer ID}=" & x
CrRpt.Preview
End Sub
```

This particular example demonstrates passing an integer variable to Seagate Crystal Reports. If however, the variable is a type string, you use slightly different code:

```
 Sub Command1_Click()
'x can be a textbox being populated with an string
Dim x as string
CrRpt.RecordSelectionFormula = "{Customer.Customer ID}=" & chr(34) & x &
chr(34)
CrRpt.Preview
End Sub
```

Any dates that are being passed to Seagate Crystal Reports must be in Seagate Crystal Reports Date format, Date(yyyy,mm,dd).

Passing a Formula at runtime

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

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

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

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

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

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

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

```
Sub Command1_Click ()
'if we hard-code the title
CrReport.FormulaFields.Item(1).text = " 'This is a Title' "
'If we use the value of the Textbox
Text1.Text = "This is a Title"
CrReport.FormulaFields.Item(1).text = chr(34) & text1.text & chr(34)
CrReport.Preview
End Sub
```

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

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

Changing a subreport's record selection formula at runtime

Changing the record selection formula for a subreport is similar to changing it for the main report. Before you can change the record selection formula for a subreport, you first must open the subreport. Please refer to *Referencing objects in a subreport, Page 87* for examples of different ways you can access a subreport.

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

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

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

Referencing a report formula using the formula name

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

You can put this code in the General Declarations.

```
Dim CrxReport As New CrystalReport1
Dim Crxformulafields As FormulaFieldDefinitions
Dim Crxformulafield As CRAXDRT.FormulaFieldDefinition
```

You can put this code in the Form_Load event procedure.

```
Private Sub Form_Load()
Set Crxrformulafields = Report.formulafields
```

Now you can cycle through the formula fields collection.

```
For Each Crxformulafield In Crxformulafields
```

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

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

Now you set the formula text.

```
Crxformulafield.Text = "Totext({Customer.Name})"
End If
Next
```

Finally, you can view the report.

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

Working with Grouping

Changing the Existing Group's Condition Field

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

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

You can place this code in the General Declarations section.

```
Dim Report As New CrystalReport1 'The existing report (ActiveX Designer)
Dim crxApp As CRAXDRT.Application
Dim crxDBField As CRAXDRT.DatabaseFieldDefinition
```

You can put this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

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

```
set crxDBField = Report.Database.Tables.Item(1).Fields.Item(12)
```

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

```
Report.Areas.Item("GH").GroupConditionField = crxDBField
```

Finally, display the report.

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

Adding a New Group to the Report

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

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

You can place this code in the General Declarations section.

```
Dim Report As New CrystalReport1 'The existing report (ActiveX Designer)
Dim crxApp As CRAXDRT.Application
Dim crxDBField As CRAXDRT.DatabaseFieldDefinition
```

You can place this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

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

```
set crxDBField = Report.Database.Tables.Item(1).Fields.Item(12)
```

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

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

```
Report.AddGroup 0, crxDBField, crGCAnyValue, crAscendingOrder
```

Finally, display the report.

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

Adding a Group Sort Field

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

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

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

1 Place this code in the General Declarations section.

```
Dim Report As New CrystalReport1 'The existing report (ActiveX Designer)
Dim crxApp As CRAXDRT.Application
Dim crxSortFields As CRAXDRT.SortFields
Dim crxSummaryField As CRAXDRT.SummaryFieldDefinition
```

2 Place this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

3 This code gets the SortFields collection for the groups:

```
Set crxSortFields = Report.GroupSortFields
```

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

```
Set crxSummaryField = Report.SummaryFields.Item(1)
```

5 Now add the Group Sort Field.

```
crxSortFields.Add crxSummaryField, crDescendingOrder
```

6 Finally, display the report.

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

Modifying a group's sort direction at runtime

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

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

1 Put this code in the General Declarations section.

```
Dim Report As New CrystalReport1
```

2 Put this code in the FORM_LOAD event procedure.

```
Private Sub Form_Load()
```

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

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

3 Finally, view the report.

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

Working with Parameter Fields

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

This code can go in the General Declarations Section.

```
Private Sub Command1_Click()
Dim crpParamDefs As CRAXDRT.ParameterFieldDefinitions
Dim crpParamDef As CRAXDRT.ParameterFieldDefinition
Dim crpSubreport As CRAXDRT.Report
Set crpParamDefs = Report.ParameterFields
```

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

```
For Each crpParamDef In crpParamDefs
With crpParamDef
Select Case .ParameterFieldName
```

It finds and sets the appropriate Crystal parameter.

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

Now it finds and sets the appropriate stored procedure parameter.

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

It opens the appropriate subreport.

```
Set crpSubreport = Report.OpenSubreport("sub1")
Set crpParamDefs = crpSubreport.ParameterFields
```

It cycles through the ParameterFieldDefinitions collection for the subreport.

```
For Each crpParamDef In crpParamDefs
With crpParamDef
MsgBox .ParameterFieldName
Select Case .ParameterFieldName
```

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

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

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

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

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

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

Working with OLE objects

Setting the location of an OLE object

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

- a bitmap,
- part of an Excel worksheet, and
- part of a Word document,

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

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

Object to hold the bitmap:

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

Calculate an integer to pass in as the object name.

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

Take an action based on the integer passed in.

```
    Select Case iModNum
```

The bitmap object:

```
    Case 1
```

Set the variable to a bitmap.

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

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

Set the bitmap on the Report equal to the variable.

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

Convert from HiMetric to twips.

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

The Excel Worksheet object:

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

The Word Document object:

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

Working with Parameter Fields

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

Thic code can go in the General Declarations Section.

```
   Private Sub Command1_Click()
   Dim crpParamDefs As CRAXDRT.ParameterFieldDefinitions
   Dim crpParamDef As CRAXDRT.ParameterFieldDefinition
   Dim crpSubreport As CRAXDRT.Report
   Set crpParamDefs = Report.ParameterFields
```

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

```
   For Each crpParamDef In crpParamDefs
   With crpParamDef
   Select Case .ParameterFieldName
```

It finds and sets the appropriate Crystal parameter.

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

Now it finds and sets the appropriate stored procedure parameter.

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

It opens the appropriate subreport.

```
Set crpSubreport = Report.OpenSubreport("sub1")
Set crpParamDefs = crpSubreport.ParameterFields
```

It cycles through the ParameterFieldDefinitions collection for the subreport.

```
For Each crpParamDef In crpParamDefs
With crpParamDef
MsgBox .ParameterFieldName
Select Case .ParameterFieldName
```

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

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

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

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

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

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

Working with Sorting

Changing the Existing Sort's Condition Field

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

```
'General Declarations
Dim Report As New CrystalReport1
Dim crxDatabaseField As craxdrt.DatabaseFieldDefinition
Private Sub Form_Load()
```

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

```
Set crxDatabaseField = Report.Database.Tables.Item(1).Fields.Item(8)
```

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

```
Report.RecordSortFields.Item(1).Field = crxDatabaseField
CRViewer1.ReportSource = Report
CRViewer1.ViewReport
End Sub
```

Adding a New Sort Field to a Report

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

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

```
Dim Report As New CrystalReport1
Dim crxDatabaseField As craxdrt.DatabaseFieldDefinition
Private Sub Form_Load()
```

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

```
Set crpDatabaseField = Report.Database.Tables.Item(1).Fields.Item(2)
```

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

```
Report.RecordSortFields.Add crxDatabaseField, crAscendingOrder
```

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

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

Working with Summary Fields

How to change summary field kind

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

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

Populate the combo box.

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

Get the SummaryFields collection.

```
Set CrystalSummaryFieldDefinitions = cr.SummaryFields
txtCount = CrystalSummaryFieldDefinitions.Count
```

Cycle through the collection.

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

Change the summary field kind.

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

Working with Text Objects

Simple and complex text objects

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

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

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

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

Changing the contents of a Text Object

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

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

Text objects are very flexible and powerful reporting tools.

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

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

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

Making a simple text change

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

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

Displaying a calculated result

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

```
Private Sub Form_Load()
```

Set up the variables.

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

Set your data source.

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

Read the records and calculate the results.

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

Format and assign the results to the text object.

```
                maxValueString = “The maximum value is “ & _
    Format(maxValue, “Currency”)
            Report.Text1.SetText maxValueString
```

Preview the report.

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

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

8 Programming the Report Viewer

What you will find in this chapter

OVERVIEW

The Report Viewer is a window you can add to your project to let your users preview their reports. The Crystal Report Viewer / ActiveX Object Model is a standard ActiveX control that can be added to any ActiveX container. In fact, it is the same ActiveX Report Viewer that you can use in Internet Explorer and Netscape for viewing reports via a browser.

You can add a viewer to your project manually, but the easiest way to do it is just to allow the RDC to add it for you automatically when you finish designing your report. If you decide to let the program do it automatically, the RDC will do the following:

- Add a form to your project.
- Add the viewer to the form.
- Generate the code for resizing the viewer whenever the form is resized.
- Provide the code that loads the viewer with the report in the RDC when you run the program.

The Report Viewer provides a robust set of properties, methods, and events that allow you to interact with the report, moving through pages, drilling down on groups, responding to user activity, and even printing your report.

- You can print a report or export it to a different file format without the use of the Report Viewer.
- If you want your users to be able to review the report on screen, however, you will need to include the Report Viewer in your project.

The Report Viewer has been enhanced substantially in Version 8:

- The Report Viewer now uses multi-threading. As a result, your users can now begin viewing a report sooner, even if the report requires that it all be run before certain values are generated (page numbering, for example, of the style “page 24 of 125”). In such a case, the Report Engine uses place holders for the yet-to-be-generated total page count. When that page count is completed, the Report Engine inserts the missing data into the pages already read.
- The report’s group tree is now loaded on-demand. This allows your users to use the tree functionality for navigation even when only a partial group tree has been loaded.
- You can now specify a page number to go to in the report you are currently viewing.
- Now you can use the Select Expert and the Search Expert in the viewer to select records and search for specific values using formulas.
- The Report Viewer now supports the use of rotated text in the report.
- The toolbar for the Report Viewer for ActiveX now has a stylish flattened look.
- You can customize the Report Viewer by resizing sections of the toolbar, adding custom bitmaps, and more.

- There is now a Help button implemented for applications. Clicking on the Help button can fire an event to your application so it can display the appropriate help.
- There are now over 30 events giving you the ability to make previewing the report a truly interactive activity.

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

Adding the Report Viewer to your project automatically

If you design your report in the Create Report Expert in the RDC, you will be asked when you're done if you want the program to automatically add the Report Viewer Component to your Visual Basic project. When you add the Report Viewer, a new Form is added to your project, and the Report Viewer is added to the form as an object.

When you click Finish in the Create Report Expert, the Crystal Report Expert dialog box appears.

1. The first option is whether or not you want a form containing the Crystal Reports Report Viewer added to your project. The Report Viewer is an ActiveX control that allows you to display reports directly in a form inside your application. Click Yes, if this option is not already selected.
2. The second option is whether or not the form containing the Report Viewer should be the first form displayed when your application runs, the startup form. Click Yes if this is the option you wish. Otherwise, you could supply the code elsewhere in your project to display the Report Viewer.
3. Click OK in the Crystal Report Expert dialog box. The program adds the form and the Report Viewer to your project.

Adding the Report Viewer to your project manually

If you choose not to let the program add the Report Viewer automatically and later decide to add it manually, here are the steps you need to follow.

1. Add a form to your project.
2. Choose Components from the Project menu.
3. Find the Crystal Report Viewer Control option and select it if it hasn't already been selected.
4. Open the Visual Basic Toolbox if it isn't already open. The Report Viewer icon should appear at the bottom of the tools. If you hold the mouse cursor over it, a Tooltip will identify it as CRViewer.
5. Double-click the Report Viewer icon and Visual Basic will add it to your form.
6. Once you have it on the form, you will need to enter the following code:

Put this code in the Form_Load event procedure to identify the report source and display the report:

```
Screen.MousePointer = vbHourglass
CRViewer1.ReportSource = Report
CRViewer1.ViewReport
Screen.MousePointer = vbDefault
```

Put this code in the Form_Resize event procedure to resize the Report Viewer to match the size of the form whenever the form is resized:

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

9 Migrating to the RDC from the OCX

What you will find in this chapter

OVERVIEW

Since it was first included in Microsoft Visual Basic, Seagate Crystal Reports has become the world standard for desktop reporting and design. It has kept pace with technological advancements by giving Visual Basic developers new ways to integrate reporting into database applications. For example, the Crystal ActiveX Control (OCX)—the tool most Visual Basic developers are familiar with—was first introduced in 1995 with Seagate Crystal Reports 4.5. In June 1998, the Report Designer Component (RDC) was launched. It's a revolutionary tool designed specifically for Visual Basic developers to create, view and modify reports within the Visual Basic Integrated Development Environment (IDE).

The purpose of this chapter is to illustrate the benefits of using the RDC for integrating reporting functionality into your Visual Basic applications. Learn how to migrate applications from the OCX to the RDC to take advantage of the latest features within Seagate Crystal Reports. Also included is an overview of the RDC, its major components, object model, and a description of its advanced features not available within the OCX.

Summary

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

- Integrates directly into the Visual Basic IDE.
- Allows you to create, view, and modify reports using Reports Experts and familiar Visual Basic code.
- Exposes all Print Engine features and provides the greatest number of events and objects to write code to.
- Performs better because it is a dual interface component with no wrapper around the Print Engine.
- Takes advantage of code completion features that are easy to use in the Visual Basic editor.
- Is fully compatible with Microsoft Visual Basic 5.0 and 6.0.

OCX

As mentioned earlier, OCX is the development interface most Visual Basic developers are familiar with because it's been a part of Seagate Crystal Reports since 1995. The OCX is based on an older version of ActiveX technology. All of its properties and methods are accessed through a single control. This limits your control of a report because it exposes only a subset of the Crystal Report Print Engine's functionality.

In addition, because the OCX acts a wrapper around the Print Engine, it's less efficient when loading a report because it can't directly access the Print Engine.

Code Comparison between the OCX and RDC

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

You can benefit from using the RDC by getting increased control over reports, such as:

- Flexible formatting like passing text to a text object.
- Enhanced printer control.
- Creating report templates with unbound fields and then binding the fields to a data source (or one of several data sources) at runtime.
- Report Variables that enable you to maintain state even if a report section needs to fire multiple times for a given record.
- The Create API which lets you create new objects, and even new reports, at runtime using code.
- The latest Print Engine features such as mapping and multiple parameters.
- And most importantly, the ability to create, view and modify reports inside the Visual Basic IDE.

OCX and RDC sample application comparison

Following are two applications that provide similar functionality—the first is created using the OCX, the second uses the RDC. The RDC example shows how to create a new application or convert an existing OCX application. With very few exceptions, you can duplicate any properties and methods set by the OCX using the RDC. Its properties and methods are very similar to the OCX, greatly reducing the time it takes you to learn the product.

The major differences between the two applications include:

- Setting the Crystal-related Project References and Components.
- Setting or accessing objects to get to the properties or methods needed for the report.
- The addition of the Crystal Report Viewer for viewing reports.

Sample Application General Description:

A report with a subreport is created off the xtreme.mdb database.

- The main report contains the Customer table and a parameter field.
- The subreport contains the Orders table and a formula field.

The OCX application consists of a Form with three Command Buttons and the OCX control.

Form Load:

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

Command1

The report is previewed to screen

Command2

- The printer is selected
- The report is printed

Command3

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

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

OCX sample application

Project | References:

No Crystal References required

Project | Components:

Crystal Report Control

Form1

```
Private Sub Form_Load()
```

Open the report.

```
CrystalReport1.ReportFileName = App.Path & "/OCX_to_RDC.rpt"
```

Change the location of the database.

```
CrystalReport1.DataFiles(0) = App.Path & "/xtreme.mdb"
```

Pass the parameter value to the main report.

```
CrystalReport1.ParameterFields(0) = "Param1;Main Report Param;True"
```

Pass the selection formula to the main report.

```
CrystalReport1.ReplaceSelectionFormula _
"{Customer.Last Year s Sales} < 50000.00"
```

Open the subreport.

```
CrystalReport1.SubreportToChange = "Sub1"
```

Change the location of the database in the subreport.

```
CrystalReport1.DataFiles(0) = App.Path & "/xtreme.mdb"
```

Pass the formula to the subreport.

```
CrystalReport1.Formulas(0) = "Formula1= " & "'Subreport Formula'"
```

Set CrystalReport1 back to using the main report.

```
CrystalReport1.SubreportToChange = ""
End Sub
Private Sub Command1_Click()
```

Set the destination to window.

```
CrystalReport1.Destination = crptToWindow
```

Preview the Report.

```
CrystalReport1.Action = 1
End Sub
Private Sub Command2_Click()
```

Set the printer driver.

```
CrystalReport1.PrinterDriver = “HPPCL5MS.DRV”
```

Set the printer port.

```
CrystalReport1.PrinterName = “HP LaserJet 4m Plus”
```

Set the printer name.

```
CrystalReport1.PrinterPort = “\\Vanprt\v1-1mpls-ts”
```

Set the destination to printer.

```
CrystalReport1.Destination = crptToPrinter
```

Print the report.

```
CrystalReport1.Action = 1
End Sub
Private Sub Command3_Click()
```

Set the Report to be exported to Rich Text Format.

```
CrystalReport1.PrintFileType = crptRTF
```

Set the Destination to Disk.

```
CrystalReport1.Destination = crptToFile
```

Set the path and name of the exported document.

```
CrystalReport1.PrintFileName = App.Path & "\OCXExport.rtf"
```

Export the report.

```
CrystalReport1.Action = 1
End Sub
```

RDC sample application

To migrate this application to the RDC, remove the OCX component from Form1, and remove the Crystal Report Control from the Project | Components menu, in addition to the steps below:

Project | References

Reference the Crystal Report 8 ActiveX Designer Runtime Library

Project | Components

Crystal Report Viewer Control

Add a second form

Add the Crystal Report Viewer Control to Form2

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

The RDC will open a standard Crystal Report (.RPT). The Report could have been Imported into or recreated in the RDC ActiveX Designer (.DSR). See details on opening a .DSR file using the RDC object model by searching for the ReportFileName property for OCX in the Report Designer Object Model reference section of the Developer Help file.

Form1:

Declare the application object used to open the rpt file.

```
Dim crxApplication As New CRAXDRT.Application
```

Declare the report object.

```
Public Report As CRAXDRT.Report
Private Sub Form_Load()
```

Declare a DatabaseTable object for setting the location of the ‘database. This object will be used for the main and subreport.

```
Dim crxDatabaseTable As CRAXDRT.DatabaseTable
```

Declare a ParameterFieldDefinition object for passing parameters.

```
Dim crxParameterField As CRAXDRT.ParameterFieldDefinition
```

Declare a Report object to set to the subeport.

```
Dim crxSubreport As CRAXDRT.Report
```

Declare a FormulaFieldDefinition object for passing formulas.

```
Dim crxFormulaField As CRAXDRT.FormulaFieldDefinition
```

Open the report.

```
Set Report = crxApplication.OpenReport _
(App.Path & "/OCX_to_RDC.rpt", 1)
```

Use a For Each loop to change the location of each DatabaseTable in the Reports DatabaseTable Collection.

```
For Each crxDatabaseTable In Report.Database.Tables
    crxDatabaseTable.Location = App.Path & "/xtreme.mdb"
Next crxDatabaseTable
```

Set crxParameterField to the first parameter in the parameterfields collection of the main report.

```
Set crxParameterField = Report.ParameterFields.Item(1)
```

Pass the value to the main report.

```
crxParameterField.AddCurrentValue "Main Report Parameter"
```

Set crxSubreport to the subreport 'Sub1' of the main report. The subreport name needs to be known to use this method.

```
Set crxSubreport = Report.OpenSubreport("Sub1")
```

Use a For Each loop to change the location of each DatabaseTable in the Subreport Database Table Collection.

```
For Each crxDatabaseTable In crxSubreport.Database.Tables
    crxDatabaseTable.Location = App.Path & "/xtreme.mdb"
Next crxDatabaseTable
```

Set crxFormulaField to the first formula in the formulafields collection of the subreport.

```
Set crxFormulaField = crxSubreport.FormulaFields.Item(1)
```

Pass the formula to the subreport.

```
crxFormulaField.Text = "'Subreport Formula'"
End Sub
Private Sub Command1_Click()
```

Call Form2 to preview the Report.

```
Form2.Show
End Sub
Private Sub Command2_Click()
```

Select the printer for the report passing the Printer Driver, Printer Name and Printer Port.

```
Report.SelectPrinter “HPPCL5MS.DRV”, “HP LaserJet 4m Plus”, “\\Vanprt\v1-
1mpls-ts”
```

Print the Report without prompting the user.

```
Report.PrintOut False
End Sub
Private Sub Command3_Click()
```

Declare an ExportOptions Object.

```
Dim crxExportOptions As CRAXDRT.ExportOptions
```

Set crxExportOptions to the Report object's ExportOptions.

```
Set crxExportOptions = Report.ExportOptions
```

Set the report to be exported to Rich Text Format.

```
crxExportOptions.FormatType = crEFTRichText
```

Set the destination type to disk.

```
crxExportOptions.DestinationType = crEDTDiskFile
```

Set the path and name of the exported document.

```
crxExportOptions.DiskFileName = App.Path & "/RDCExport.rtf"
```

Export the report without prompting the user.

```
Report.Export False
End Sub
```

Form2:

```
Private Sub Form_Load()
```

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

```
CRViewer1.ReportSource = Form1.Report
'View the Report
CRViewer1.ViewReport
End Sub
Private Sub Form_Resize()
```

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

```
CRViewer1.Top = 0
CRViewer1.Left = 0
CRViewer1.Height = ScaleHeight
CRViewer1.Width = ScaleWidth
End Sub
```

Note: For a more comprehensive OCX/RDC comparison, see the Developer's online help.

10 Working with Visual C++, Visual InterDev and Lotus Domino

What you will find in this chapter

OVERVIEW

While much of this Developer's Guide is aimed at the Visual Basic developer, it is important to note that the integration methods can be used in any environment that supports COM. This chapter shows you how to use different integration methods with other popular development environments.

USING THE RDC WITH VISUAL C++

There are different ways to access the Report Designer Component (RDC) and any other COM Automation server through Visual C++. This section describes how to use the #import method.

Manipulating the RDC in Microsoft Visual C++ involves three steps:

1. Define and instantiate a variable to be used to manipulate the RDC COM object.
2. Instantiate an actual instance of the RDC COM object and assign it to a variable.
3. Manipulate the Properties and Methods and then output the report.

The following section leads you through the details in each of these steps.

Printing a Report through Visual C++

1 Open Visual C++ 6.0 if it isn't already running.

2 From the File menu, select NEW. In the New dialog box select the MFC AppWizard (exe) from the Projects tab. Type in MyRDC for the Project Name and click OK.

3 Click the FINISH button to accept the defaults for the MFC AppWizard and then click OK in the New Project Information dialog box.

4 Add a reference to the RDC runtime object Model. From the File | Open menu select the StdAfx.h file that was generated by the MFC App wizard and Click OPEN. Add the following line after the #include directives:

```
#import "C:\Program Files\Seagate Software\Crystal Reports\Developer
Files\include\craxdrt.tlb"
```

Note: This is the default location of the RDC runtime object model (craxdrt.dll).

5 Before you can invoke an RDC object, you must initialize OLE. From the File | Open menu, open the MyRDC.CPP file and add the following code:

```
struct InitOle {
InitOle()  { ::CoInitialize(NULL); }
~InitOle() { ::CoUninitialize();   }
} _init_InitOle_;
```

6 For optional Variant parameters declare a dummy variable in the MyRDC2.CPP class.

```
Variant dummy;
```

7 Add the following constants in the declaration's of the MyRDC2.CPP class:

```
// The constants needed to create the Application and Report Objects COM
objects

const CLSID CLSID_Application =
{0xb4741fd0,0x45a6,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};

const IID IID_IApplication =
{0x0bac5cf2,0x44c9,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};

const CLSID CLSID_ReportObjects =
{0xb4741e60,0x45a6,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};

const IID IID_IReportObjects =
{0x0bac59b2,0x44c9,0x11d1,{0xab,0xec,0x00,0xa0,0xc9,0x27,0x4b,0x91}};
```

Opening and Printing a Crystal Report through the RDC:

1 From the File | Open menu open the MyRDC.CPP file. Add the following code to the MyRDC.InitInstance() method just before the return statement:

```
// A dummy variant

VariantInit (&dummy);

dummy.vt = VT_EMPTY;

HRESULT                 hr = S_OK;

                IApplicationPtr m_Application = NULL;

                IReportPtr m_Report = NULL;


                // Specify the path to the report you want to print

_bstr_t ReportPath("c:\\Program Files\\Seagate Software\\Crystal
Reports\\Samples\\Reports\\General Business\\Inventory.rpt");

_variant_t vtEmpty(DISP_E_PARAMNOTFOUND, VT_ERROR);

// Instantiate the IApplication object

hr = CoCreateInstance(CLSID_Application, NULL, CLSCTX_INPROC_SERVER ,
IID_IApplication, (void **) & CRAXDRT::IApplication);

//Open the Report using the OpenReport method

m_Report = m_Application->OpenReport(ReportPath, dummy)

//Print the Report to printer

m_Report->PrintOut(dummy, dummy, dummy, dummy
);
```

2 Finally, from the Build menu, select Rebuild All.

That's all there is to it. There are hundreds of properties, methods, and events in the Report Designer Component object model that you can manipulate through your code to meet the demands of virtually any reporting requirement.

USING THE RDC WITH VISUAL INTERDEV

The information below is directed to the developer of Web applications using the Report Integration Controls.

Installation

When you install Seagate Crystal Reports, by default the Report Integration Controls are installed in the folder SEAGATE SOFTWARE\SHARED\DESIGN TIME CONTROL. Initially, to set up the Report Integration Controls as Design Time Controls (DTC's) for use in Visual InterDev 6.0, you need to do the following:

1 Right-click in the Visual InterDev Toolbox. Choose CUSTOMIZE TOOLBOX from the shortcut menu.

2 In the Customize Toolbox dialog box, select the Design Time Controls tab. Check the two items ReportSource and ReportViewer.

The Report Integration Controls are comprised of the two DTC's, ReportSource and ReportViewer. You will find that these two DTC's are added to the Visual InterDev Toolbox. We will call them Report Source Control and Report Viewer Control below.

Adding Reports to the Current Project

You must first add a report to the current Visual InterDev project before you can apply any DTC's to it.

Insert a Report Source Control (DTC)

When you insert a Report Source Control into your Web application, you actually create an instance of the Report Source Control. Its name is ReportSourcei, where i is the i-th instance of the Report Source Control, i=1,2,3…

When you right-click on an instance of a Report Source Control, you will invoke the Report Source Properties dialog box.

The properties you can specify for an instance of a Report Source Control include the following:

Attach a report to the Report Source Control

General information: In the General tab of the Report Source Properties dialog box, you may attach a report to this instance of the Report Source Control from the list of reports included in the current project.

Logon and test connectivity

Logon information for database: In the Accounts tab of the Report Source Properties dialog box, you can specify the server and database names for each of the tables used in the attached report, and test connectivity for each of these servers after entering the logon information.

If the report is hosted on a Web Reports Server, you may leave the logon information blank, which will in turn prompt the end user to enter the logon information. If the report is hosted on an ASP server, then you must enter the logon information.

Specify how to handle parameters

Parameters: In the Parameters tab of the Report Source Properties dialog box, if the report is hosted on a Web Reports Server, you may specify whether to prompt the end user to choose from a default list of values, or enter a value for a parameter.

However, if the report is hosted on an ASP server, then you must choose a value for each input parameter.

Specify selection formulas

Selection formula: In the Formula tab of the Report Source Properties dialog box, you may specify your own selections formulas, or to modify existing selection formulas.

Insert a Report Viewer control (DTC)

When you insert a Report Viewer Control into your Web application, you create an instance of the Report Viewer Control. Its name is ReportVieweri, where i is the i-th instance of the Report Viewer Control, i=1,2,3…

When you right-click on an instance of a Report Viewer Control, you will invoke the Report Viewer Properties dialog box.

Point the Viewer at the Report Source Control

The properties you can specify for an instance of a Report Viewer Control include:

General information: In the General tab of the Report Viewer Properties dialog box, you can select an instance of a Report Source Control and attach it to this instance of the Report Viewer Control. Upon clicking the Advanced button, you can specify a virtual path which is an alias to the directory of the Report Viewer Control.

Specify options for end users

Options for end users: In the Options tab of the Report Viewer Properties dialog box, you may specify the following for the end user:

- The Report Viewer to use, whether it is the Report Viewer for Java or Report Viewer for ActiveX
- The language to use for the Report Viewer, depending on the fonts installed on the current system
- The size of the Report Viewer
- The ability to refresh report in the Report Viewer
- The ability to print report in the Report Viewer
- The ability to export report from the Report Viewer
- The ability to search in report in the Report Viewer
- The ability to drill down in report in the Report Viewer

- The ability to prompt on refresh of report in the Report Viewer
- The ability to generate group tree in the Report Viewer
- The ability to display group tree in the Report Viewer

When you click OK, you're finished.

WORKING WITH LOTUS DOMINO

You can easily add powerful reporting capabilities to your Lotus Domino applications with Seagate Crystal Reports. You can integrate the reporting functions in one of two ways: using the Automation Server or the ActiveX Control.

Automation Server

When you are using the Automation Server, you don't have to add any objects to your application. You must have Seagate Crystal Reports on the same machine, however, when you are creating the application.

To use the Automation Server method:

1 Install Seagate Crystal Reports.

2 Create your report and save it. For information on creating reports, see the *Seagate Crystal Reports User's Guide*.

3 In Domino Designer, create an action button in a view or form.

4 Enter code similar to the following in the "click" portion for the action.

Action Button – Click

```
Sub Click(Source As Button)
     Set app = CreateObject("crystal.crpe.application")
     Set rep = app.Openreport("C:\program files\seagate software\seagate
analysis\samples\Lotus Domino\wwsales.rpt")
     rep.database.tables.Item(1).SetLogoninfo "Local", "craze", "John Doe/
seagatesoftware", "password2"
     rep.PrintWindowOptions.CanDrillDown = True
     rep.PrintWindowOptions.HasRefreshButton = True
     rep.PrintWindowOptions.HasGroupTree = True
     rep.PrintWindowOptions.HasCloseButton = True
     rep.Preview
End Sub
```

Active X (OCX) Control

Using this method, you will need to add the ActiveX control to your form or view.

1. Install Seagate Crystal Reports.
2. Create your report and save it. For information on creating reports, see the *Seagate Crystal Reports User's Guide*.
3. Add the ActiveX Control to the form or view. (For directions on how to do this in the Domino environment, please refer to Domino online help topic "Using Lotus Components".
4. To declare the control, add the following line to the global declarations section for the form.

   Global FORM Declarations:

   ```
   Dim crystalReport
   ```

5. Now create an action button and put code similar to the following in the "click" section.

   Action Button – Click:

   ```
   Sub Click(Source As Button)
        Dim workspace As New NotesUIWorkspace
        Dim Session As New NotesSession
        Set db = Session.CurrentDatabase
        Set uidoc=workspace.CurrentDocument
        Set doc = uidoc.Document
        Set crystalReport = doc.EmbeddedObjects(0)
        Set handle = crystalReport.Activate(False)
             handle.connect = "dsn=;uid= John Doe/seagatesoftware;pwd=
   password1;dsq="
        handle.ReportFileName = "C:\program files\seagate software\seagate
   analysis\samples\lotus domino\wwsales.rpt"
        handle.action = 1
        End Sub
   ```

   Note: In both of these examples, the ReportFileName points to the file location of the report. You will need to change this to match the actual report file location on your drive of course.

I N D E X

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W