Business Objects

Crystal Reports XI Release 2

Migrating Applications from the RDC to the .NET assemblies

Overview

This document is a guide to migrating Crystal Reports 8.5 Report Designer Component (RDC) Visual Basic (VB) applications to the Crystal Reports XI Release 2 .NET assemblies. Specifically, an overview of the benefits of upgrading, the Crystal Reports .NET object model, and the key considerations when migrating to .NET, as well as a comparison of RDC and the equivalent .NET code are discussed.

Contents

Introduction

The purpose of this document is to help you migrate your RDC applications to the Crystal Reports XI Release 2 .NET assemblies. Topics include the benefits of upgrading, the Crystal Reports .NET object model, and the key considerations when migrating to .NET, as well as a comparison of RDC and the equivalent .NET code.

History

The Crystal Reports Report Designer Component (RDC) was created in 1998 to enable developers to create, view, and modify reports within the Microsoft Visual Basic (VB) integrated development environment (IDE). The RDC is integrated into the VB IDE using the Component Object Model (COM). COM has been superseded by Microsoft .NET and the RDC has been replaced by the Crystal Reports .NET assemblies. The .NET assemblies are bundled with Microsoft Visual Studio .NET 2002 and later versions, and are now also fully integrated within Borland C# Builder 2005 and Borland Delphi 2005.

Benefits of upgrading to Crystal Reports .NET assemblies

Crystal Reports XI introduces new features that improve productivity, for example, reducing the effort required to view, print, and export reports. Significant enhancements have been made to the existing feature set, including faster report design, enhanced web integration, and better performance and report management for .NET applications.

Crystal Reports XI provides the following benefits of migrating to the .NET assemblies:

- **Easy report building from Datasets and XML.** Datasets built using ADO.NET are fully supported. One advantage of the Dataset approach is that the same Dataset used for display in view or edit mode can be used to create the report.
- **Native support for ASP.NET and Webform.** The Crystal Reports Webform viewer control enables you to take advantage of all the features of ASP.NET.
- **Flexible language support.** Create reporting applications in VB.NET, C#, J#, or any other language supported in Crystal Reports.
- **Automatic code generation for .NET.** When you create a .NET project, VB.NET and C# code is automatically generated to connect the engine and viewers.
- **Simplified deployment of reporting applications.** A single integrated merge module contains all the necessary assemblies to deploy an application. Properties in the merge module can be customized to determine which libraries are deployed. If Crystal Reports is included in the application, the merge module is automatically detected when generating a setup project.

Full side-by-side deployment is supported. Different versions of the .NET run-time files can co-exist without DLL conflicts.

- **Effortless application growth.** A project migration wizard is available to upgrade applications from older versions of Crystal Reports to version XI Release 2. The migration wizard optionally updates .NET project references to the latest version.

  As the number of users accessing your .NET application increases, you can modify the application to use Crystal Reports Server or BusinessObjects Enterprise with minimal code changes. Connecting .NET view-only applications to Crystal Reports Server or BusinessObjects Enterprise significantly improves performance.

  Crystal Reports Server and BusinessObjects Enterprise include report management, scheduling, security, and web distribution capabilities. All the APIs needed to integrate these capabilities into applications are included.

- **IDE Integration.** Crystal Reports is integrated into Microsoft Visual Studio .NET 2003 and 2005, Borland C# Builder, and Borland Delphi .NET.

In addition, Crystal Reports XI Release 2 provides the following benefits of migrating to the .NET assemblies:

- **Dynamic and cascading prompts.** You can now populate a prompt list in a report with live data from a database rather than maintaining static prompt lists in individual reports. A single prompt definition can be stored in a repository and shared among multiple reports, improving both run-time scalability and design-time productivity.
- **Dynamic image location.** Images are added and dynamically updated at report run-time. Pictures and graphics can now be placed in a report through a link in the database so that it is no longer necessary to store images within the database. This supports the common practice of storing images on the web server and storing references to those images in the database.
- **Intelligent charting.** New drag-and-drop charting and **c**ross-tabs provide intelligent charting. Variables are approximated when a chart is dropped into a section. Chart design is faster and easier because charts are updated automatically when new variables are added.
- **HTML preview.** Report authors can see how reports will look when published to the web. This is part of a streamlined iterative report design/view process.
- **Additional export formats.** You can export reports to a number of popular formats, such as Microsoft Excel and Word, Adobe PDF, XML (rich-client only), HTML, ODBC, and common data interchange formats. Distribution of information becomes easier and allows you to use familiar tools to manipulate the report data.

- **Editable RTF format.** Reports can be delivered to end-users in a RTF format that allows them to make their own modifications in their favorite word processing application. This format is ideal for ease of editing.
- **Additional data connectivity.** You can connect to almost any data source, including Enterprise data sources (Oracle®, IBM® DB2®, Sybase®, Informix®), XML, Teradata and Exchange 2003, ODBC, OLEDB, ADO.NET. Better data connectivity to Microsoft Exchange allows tighter integration with Microsoft products.

The complete list of Crystal Reports features is available at this location:

http://www.businessobjects.com/products/reporting/crystalreports/features.asp

Porting an RDC application to the Crystal Reports .NET assemblies

In this section, the process of porting an RDC application to the .NET assemblies is discussed. A brief description of the sample application is provided, followed by a description of the RDC implementation. Next, the steps required to port this VB 6 RDC application to the .NET assemblies are covered, followed by an examination of the resulting .NET implementation. To download the sample application code, refer to Finding more information.

General description of the example application

A report with a subreport is created using 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 application consists of two forms. The first form, frmMain, loads the report and provides three command buttons to preview, print, and export the report. The second form, frmPreview, contains the Report Viewer, for previewing the report.

Below is a summary of the two forms:

frmMain

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 on the screen.

Command2

- The printer is selected.
- The report is printed.

Command3

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

frmPreview

Report View

- The source of the Report Viewer is set to the main report.
- The report is viewed.

Form Resize

- The Report Viewer is resized to the dimensions of the preview form.

The RDC implementation

The following section examines a VB 6 application that uses the Crystal Reports 8.5 RDC.

Project settings

Project > References menu

- Crystal Report 8.5 ActiveX Designer Runtime Library

Project > Components menu

- Crystal Report Viewer Control

frmMain

```
'Declare the application object used to open the rpt file
Dim crxApplication As New CRAXDRT.Application
'Declare the report object
Public Report As CRAXDRT.Report

Private Sub Form_Load()

'Declare a DatabaseTable Object
Dim crxDatabaseTable As CRAXDRT.DatabaseTable
'Declare a Report object to set to the subreport
Dim crxSubreport As CRAXDRT.Report
```

```
'Open the report
Set Report = crxApplication.OpenReport _
  (App.Path & "\RDC_To_CR.Net.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

'Pass the Parameter value to the first parameter field in
'the ParameterFields collection of the Report
Report.ParameterFields.Item(1) _
  .AddCurrentValue "Main Report Parameter"

'Set crxSubreport to the subreport 'Orders' of the main
'report. The subreport name needs to be known
'to use this method.
Set crxSubreport = Report.OpenSubreport("Orders")

'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

'Pass the formula's text to the first formula field
'in the FormulaFields collection of the subreport
crxSubreport.FormulaFields.Item(1).Text = _
  "'Subreport Formula'"

End Sub

Private Sub cmdPreview_Click()
  'Call frmPreview to preview the Report
  frmPreview.Show
End Sub

Private Sub cmdPrint_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\v1-1mp1s-ts"
  'Print the Report without prompting user.
  Report.PrintOut False
End Sub

Private Sub cmdExport_Click()

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

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

'Set the path and name of the exported document
Report.ExportOptions.DiskFileName = _
  App.Path & "\RDCExport.rtf"

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

End Sub

Private Sub Form_Unload(Cancel As Integer)
  Set crxApplication = Nothing
  Set Report = Nothing
End Sub
```

frmPreview

```
Private Sub Form_Load()

  'Set the report source for the Report Viewer
  CRViewer1.ReportSource = frmMain.Report

  'View the Report
  CRViewer1.ViewReport

End Sub
```

```
Private Sub Form_Resize()
  'This code resizes the report Viewer control to
  'frmPreview's dimensions
  CRViewer1.Top = 0
  CRViewer1.Left = 0
  CRViewer1.Height = ScaleHeight
  CRViewer1.Width = ScaleWidth
End Sub
```

Porting the RDC application

Porting the RDC application to the .NET assemblies involves the following:

- Upgrading the project from VB 6 to VB.NET.
- Changing the project settings.
- Adding the .NET Report Viewer to the preview form.
- Migrating the application code to the new .NET API.

Upgrading to Visual Basic.NET

Microsoft Visual Studio .NET provides support for upgrading legacy VB projects. To upgrade the VB 6 RDC project to VB.NET, complete these steps:

1. Open the RDC project in Visual Studio .NET. The Visual Basic Upgrade Wizard appears.
2. Follow the steps in the Visual Basic Upgrade Wizard.

The RDC project is now converted to VB.NET.

For more information on the Visual Basic Upgrade Wizard, see *Using the Visual Basic Upgrade Wizard* at

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vstchexpvsnetlab5.asp

Changing the project settings

Remove the following references from the **Project > References** menu:

- AxCRVIEWERLib
- CRAXDRT
- CRVIEWERLib

Adding the .NET Report Viewer

Add the Report Viewer to the preview form using the **Windows Forms** toolbox. Verify that the following references are automatically added to the **Project > References** menu:

- CrystalDecisions.Windows.Forms
- CrystalDecisions.Shared
- CrystalDecisions.CrystalReports.Engine

NOTE	Adding the Report Viewer to the preview form also results in a reference to CrystalDecisions.ReportSource on the **Project** > **References** menu. This reference is not required for the example application.

Migrating code to the new Crystal Reports .NET API

Modifying the code to use the new .NET API requires knowledge of the Crystal Reports .NET object model. For a description of the object model, refer to the section Working with the object models in the .NET SDK. Comparative code samples for the most common properties and methods in the RDC are presented in Appendix A: Comparing RDC and .NET code.

Below, some of the code changes required in our example application are examined. Specifically, changes related to the following:

- Loading the report
- Printing the report
- Exporting the report

Loading the Report

In the RDC, the **Application** object (crxApplication) loads the report using the **OpenReport** method and returns a **Report** object. In the .NET assemblies, the **ReportDocument** object (Report) loads itself.

RDC code

```
'Open the report
Set Report = crxApplication.OpenReport _
  (App.Path & "\RDC_To_CR.Net.rpt", 1)
```

.NET code

```
'Open the report
Report.Load _
  (System.AppDomain.CurrentDomain.BaseDirectory & _
   "\RDC_To_CR.Net.rpt")
```

Printing the Report

In the .NET assemblies, the **PrintOptions** object is used to set options for printing a report. Options that can be set include the printer name, paper size, paper orientation, and page margins.

RDC Code

```
'Select the printer for the report passing the
'Printer dRiver, Printer Name and Printer Port
Report.SelectPrinter "HPPCL5MS.DRV", _
  "HP LaserJet 4m Plus", "\\Vanprt\v1-1mp1s-ts"
'Print the Report without prompting user
Report.PrintOut False
```

.NET code

```
'Select the printer for the report passing the Printer Port
Report.PrintOptions.PrinterName = "\\Vanprt\v1-1mp1s-ts" _
  Report.PrintToPrinter(1, False, 0, 0)
```

Exporting the Report

The .NET assemblies simplify much of the code required in your application. Instead of exposing the **`ExportOptions`** object, they provide a single method for commonly used operations, such as exporting to disk.

RDC code

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

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

'Set the path and name of the exported document
Report.ExportOptions.DiskFileName = _
  App.Path & "\RDCExport.rtf"

'Export the report without prompting the user
Report.Export False
```

.NET code

```
'Export the report to RichTextFormat on disk
Report.ExportToDisk _
  (CrystalDecisions.[Shared].ExportFormatType.RichText, _
   VB6.GetPath & "\RDCExport.rtf")
```

The .NET implementation

Now that the example application has been ported to .NET, the new implementation differs from the original RDC implementation in several ways. The major changes to the implementation include the following:

- The project settings for Crystal Reports-related project references and components.
- The new .NET API methods used to manipulate the report.

The following sections describe what you see when you examine the application in Visual Studio .NET.

Project settings

Project > References menu

- CrystalDecisions.Windows.Forms
- CrystalDecisions.Shared
- CrystalDecisions.CrystalReports.Engine

frmMain.vb

```
Option Strict Off
Option Explicit On
  Imports CrystalDecisions.CrystalReports.Engine

Friend Class frmMain
  Inherits System.Windows.Forms.Form

#Region "Windows Form Designer generated code"
…
…
#End Region

#Region "Upgrade Support "
  Private Shared m_vb6FormDefInstance As frmMain
  Private Shared m_InitializingDefInstance As Boolean
  Public Shared Property DefInstance() As frmMain
    Get
      If m_vb6FormDefInstance Is Nothing _
          OrElse m_vb6FormDefInstance.IsDisposed Then
        m_InitializingDefInstance = True
        m_vb6FormDefInstance = New frmMain()
        m_InitializingDefInstance = False
```

```
      End If
      DefInstance = m_vb6FormDefInstance
    End Get
    Set
      m_vb6FormDefInstance = Value
    End Set
  End Property
#End Region

'Declare the report object
Public Report As ReportDocument

Private Sub frmMain_Load _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs) Handles MyBase.Load

  'Declare a DatabaseTable Object
  Dim crxDatabaseTable As Table
  'Declare a Report object to set to the subreport
  Dim crxSubreport As ReportDocument

  Report = New ReportDocument
  'Open the report
  Report.Load _
    (System.AppDomain.CurrentDomain.BaseDirectory & _
    "\RDC_To_CR.Net.rpt")

  '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 = _
      System.AppDomain.CurrentDomain.BaseDirectory & _
        "\xtreme.mdb"
  Next crxDatabaseTable

  'Pass the Parameter value to the first parameter field
  'in the ParameterFields collection of the Report
  Report.ParameterFields.Item(0).CurrentValues _
    .AddValue("Main Report Parameter")
```

```
    'Set crxSubreport to the subreport 'Orders' of the main
    'report. The subreport name needs to be known to use this
    'method.
    crxSubreport = Report.OpenSubreport("Orders")

    '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 = _
        System.AppDomain.CurrentDomain.BaseDirectory & _
          "\xtreme.mdb"
    Next crxDatabaseTable

    'Pass the formula's text to the first formula field
    'in the FormulaFields collection of the subreport.
    crxSubreport.DataDefinition.FormulaFields.Item(0).Text = _
      "'Subreport Formula'"

End Sub

Private Sub cmdPreview_Click _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs) _
  Handles cmdPreview.Click
  frmPreview.DefInstance.Show()
End Sub

Private Sub cmdPrint_Click _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs)
  Handles cmdPrint.Click

  'Select the printer for the report passing the
  'Printer Name
  Report.PrintOptions.PrinterName = "\\Vanprt\v1-1mp1s-ts"
  Report.PrintToPrinter(1, False, 0, 0)
End Sub

Private Sub cmdExport_Click _
  (ByVal eventSender As System.Object, _
```

```
    ByVal eventArgs As System.EventArgs) _
  Handles cmdExport.Click

  'Export the report to RichTextFormat on disk
  Report.ExportToDisk _
    (CrystalDecisions.[Shared].ExportFormatType.RichText, _
     VB6.GetPath & "\RDCExport.rtf")
End Sub

Private Sub frmMain_Closed _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs) _
  Handles MyBase.Closed
  Report = Nothing
End Sub

End Class
```

frmPreview.vb

```
Option Strict Off
Option Explicit On
Friend Class frmPreview
  Inherits System.Windows.Forms.Form

#Region "Windows Form Designer generated code "
…
…
#End Region

#Region "Upgrade Support "
  Private Shared m_vb6FormDefInstance As frmPreview
  Private Shared m_InitializingDefInstance As Boolean
  Public Shared Property DefInstance() As frmPreview
    Get
      If m_vb6FormDefInstance Is Nothing _
         OrElse m_vb6FormDefInstance.IsDisposed Then
        m_InitializingDefInstance = True
        m_vb6FormDefInstance = New frmPreview()
        m_InitializingDefInstance = False
      End If
```

```
      DefInstance = m_vb6FormDefInstance
    End Get
    Set
      m_vb6FormDefInstance = Value
    End Set
  End Property
#End Region

Private Sub frmPreview_Load _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs) Handles MyBase.Load

  'Set the report source for the Report Viewer to the report
  CrystalReportViewer1.ReportSource = _
    frmMain.DefInstance.Report

  'View the Report
  CrystalReportViewer1.Show()
End Sub

Private Sub frmPreview_Resize _
  (ByVal eventSender As System.Object, _
   ByVal eventArgs As System.EventArgs) _
  Handles MyBase.Resize
  'This code resizes the report Viewer control to
  'frmPreview's dimensions
  CrystalReportViewer1.Top = 0
  CrystalReportViewer1.Left = 0
  CrystalReportViewer1.Height = ClientRectangle.Height
  CrystalReportViewer1.Width = ClientRectangle.Width
End Sub

End Class
```

Summary

This section demonstrated how to port a simple RDC application to the .NET assemblies. Porting the example application required upgrading the application to VB.NET, changing the project settings, and modifying the application code to use the new .NET API. For more information on how to use the Crystal Reports .NET object model, refer to the section Working with the object models in the .NET SDK.

Key considerations when migrating to .NET

This section highlights some important factors to consider when migrating from the RDC to the .NET assemblies. It also provides details of some new and changed features in the .NET assemblies that can improve your existing RDC applications.

Integrated IDE changes

For developers, migrating from RDC to the .NET assemblies is synonymous with migrating from a COM-based IDE to a .NET-based IDE. Crystal Reports is now tightly integrated into Microsoft Visual Studio .NET, Borland C#, and Borland Delphi .NET. Migrating from RDC to .NET therefore gives you a choice of IDEs.

These IDEs offer additional features, such as enhanced debugging abilities, code completion, and resource management that were previously not available in Visual Studio .NET. Visual Studio .NET provides a built-in upgrade wizard that upgrades legacy VB projects to VB.NET.

Visual Studio .NET provides an integrated Crystal Reports Designer and a project migration wizard. Existing .NET projects with integrated Crystal Reports 9 or 10 functionalities are updated to the latest version by the migration wizard.

Object model changes

The .NET assemblies provide the features needed to access and manipulate reports, report off data, create database connections, print, and set export types, options, and various report object properties.

Some RDC features that are not supported in the .NET assemblies are discussed in this section. For a detailed comparison of RDC and the .NET assemblies features, refer to Appendix B: Feature comparison of RDC and .NET assemblies.

If you require a feature that is absent in the .NET assemblies, an alternative is to use Crystal Reports Server or BusinessObjects Enterprise since they provide greater report modification capabilities. A feature comparison of all Crystal Reports versions is available at this location:

http://www.businessobjects.com/pdf/products/crystalreports/crxi_feat_ver_ed.pdf

General database features

The .NET assemblies offer a comparable set of features to those supported in the RDC, except they do not support Show SQL Query, Perform Grouping on Server, and Field Mapping.

The run-time dataset support through ADO has been deprecated. The same information is now accessed through ADO.NET.

Related .NET class: **`Database`**

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Exporting

The .NET assemblies can export to a file, Exchange folder, and Microsoft Mail. In addition, HTTP response and stream objects are new export destinations. Exporting to an application, Lotus Domino, and Lotus Domino mail is not supported.

The .NET assemblies support most of the export formats commonly used in RDC, namely PDF, Crystal Reports, HTML, Excel, Word, record style, RTF, CSV, and text. They no longer support the tab separated text, XML, Report Definition, file, and ODBC formats.

The RDC in Crystal Reports XI supports the new editable RTF format. This feature is not supported in the .NET assemblies but can be accessed using Crystal Reports Server or BusinessObjects Enterprise.

Related .NET class: **`ExportOptions`**

Namespace: **`CrystalDecisions.CrystalReports.Shared`**

Report Object common properties

The .NET assemblies support all **Report** object common properties, with the exception of **ConditionFormula**, **HyperlinkType**, **HyperlinkText**, and **LinkedURI**.

Related .NET class: **`ReportObject`**

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Specific report objects

Specific report objects supported in the .NET assemblies include **BlobFieldObject**, **BoxObject**, **ChartObject**, **CrosstabObject**, **FieldObject**, **LineObject**, **MapObject**, **OlapGridObject**, **OleObject**, **SubReportObject**, **TextObject**, and **FieldHeadingObject**.

The **Report** object is the parent class of these specific report objects, and thus all supported properties are inherited from the base **Report** object. A list of additional properties and methods supported in each of the specific report objects is found in Appendix B: Feature comparison of RDC and .NET assemblies.

Related .NET classes**: `BlobFieldObject, BoxObject, ChartObject, CrosstabObject, FieldObject, LineObject, MapObject, OlapGridObject, OleObject, SubReportObject, TextObject, FieldHeadingObject`**

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Report options

The .NET assemblies support the following report options: **SaveDataWithReport**, **SaveSummaryWithReport**, and **UseDummyData**. The latter two options are not available in the RDC.

Related .NET class: **`ReportOptions`**

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Fields

Fields supported in the ReportDocument object model include Formula, **Group Name**, **Parameter**, **RunningTotal**, **Special**, **SQLExpression**, and **Summary**. In general, the ability to create fields is not supported in the .NET assemblies. For details, refer to Appendix B: Feature comparison of RDC and .NET assemblies.

Related .NET classes: **`FormulaFieldDefinitions`**, **`GroupNameFieldDefinitions`**, **`ParameterFieldDefinitions`**, **`RunningTotalFieldDefinitions`**, **`SpecialVarFieldDefinitions`**, **`SQLExpressionFieldDefinitions`**, **`SummaryFieldDefinitions`**.

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Groups

The .NET assemblies provide partial support for groups; specifically they support the ability to read **ConditionFields**. Only a subset of the **GroupOptions** functionality is supported, including **sortDirection**, **NumberOfTopOrBottomNGroups**, **discardOtherGroups**, and **notInTopOrBottomNName**.

Related .NET Class: **`Group`**, **`GroupOptions`**

Namespace: **`CrystalDecisions.CrystalReports.Engine`**

Business Views

The .NET assemblies do not support reporting from Business Views.

Report alerts

The ability of the RDC to programmatically set and get report alerts is not supported. If your application requires run-time manipulation of report alerts, the Report Application Server (available with Crystal Reports Server and BusinessObjects Enterprise) has the appropriate API for report alerts.

Drill-down events

The RDC can listen to **Map** object drill-down events in order to trigger other actions. Although this feature is not included in the .NET

assemblies, all other RDC drill-down features are supported. Report parts is also supported, but not by the RDC.

Search Expert

The .NET assemblies do not support any Search Expert features.

Events

The .NET assemblies do not support the following events: **NoData**, **BeforeFormatPage**, **AfterFormatPage**, and **FieldMapping**.

NOTE	These features are exclusive to the RDC and are not available from Crystal Reports Server or BusinessObjects Enterprise.

Data access

The RDC supports data access through Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO). ADO enables you to connect to most ISAM, ODBC, and SQL data sources available to Windows applications. In addition to using data objects, the RDC makes it possible to design reports based on data definition files and database field types without including information about the actual data source. The existence of a data source at design time is no longer necessary. Data can be created dynamically at run-time and assigned to the report object of the RDC.

The .NET assemblies offer access to virtually any data with native ODBC and OLE DB connectivity to relational OLAP, XML, legacy, and enterprise data sources (including Oracle®, IBM® DB2®, Sybase®, Microsoft SQL Server, and Informix®). You can also access custom, user-defined application data by connecting to JavaBeans, ADO.NET, and COM data providers. The improved data connectivity allows you to connect to almost any data source to support your legacy applications. You can also choose the data source and data connectivity technology that best fits your needs when designing a new application.

In general, data connections for RDC reports set up at design time are not affected by the migration to .NET. One notable exception is the runtime dataset supported in ADO. The .NET assemblies do not support the ability to pass in an ADO dataset at run time. Instead, you should make use of the run-time support for ADO.NET.

Reusing UFLs

A User Function Library (UFL) is a library of user-defined functions made accessible (through a COM interface) to you in the Crystal Reports Formula Editor.

A number of user-defined functions are installed with Crystal Reports for Visual Studio .NET for use in formulas. Some of these functions are part of the main processing DLL, Crpe32.dll, while others are provided

by U2ldts.dll (which provides date and date-time conversion functions) and U2lfinra.dll (which provides financial-related functions).

If you are migrating to .NET, consider the investment made in existing managed code libraries and whether these libraries can be used in .NET.

Crystal Reports XI Release 2 enables you to reuse functions previously developed in your RDC applications. When you install Crystal Reports for .NET, the U2lcom.dll file is installed in the folder C:\Program Files\Common Files\Crystal Decisions\1.0\bin. U2lcom.dll is a UFL that can make use of COM DLL files.

UFLs are dynamic link libraries that expose functions from a COM automation server. These automation servers must be built with a COM language and the DLL must be named according to the standard naming convention defined by Crystal Reports.

Your existing ActiveX DLL should be placed in the same folder as the U2lcom.dll library included in Crystal Reports. This file is normally found in the C:\Program Files\Common Files\Crystal Decisions\1.0\Bin folder.

Register your ActiveX DLL with Windows by opening a command prompt and running the following command:

```
regsvr32.exe "C:\Program Files\Common Files\Crystal Decisions\1.0\Bin\filename.dll"
```

For more information on how to create your own user-defined function, refer to the following documents:

For Visual Basic:
http://support.businessobjects.com/communityCS/TechnicalPapers/scr_user_defined_functions.pdf

For Delphi:
http://support.businessobjects.com/communityCS/TechnicalPapers/ufl_delphi.pdf

NOTE	While the .NET assemblies in Crystal Reports XI Release 2 also provide support for creating UFLs using VB.NET and C#, they must be registered for COM interoperability before they can be used. For more information, in the Crystal Reports Help, go to Contents > Tutorials and Sample Code > Other Tutorials > Creating a User Function Library.

Deployment

The .NET assemblies in Crystal Reports XI Release 2 eliminate many issues commonly associated with deployment. Including a report in your application for deployment is now a simple process requiring only a few steps. Enhanced configurable deployment options provide full control over deployed database and exporting DLLs, as well as a configurable option for when the crystalreportviewers115 virtual directory is created.

Full side-by-side deployment is supported; therefore, different versions of the .NET run-time files can co-exist without DLL conflicts.

When you deploy a .NET application, do the following:

- Distribute your report files.
- Deploy the Crystal Reports merge module.
- Install the .NET framework on the client computer.

NOTE	The Developer editions of Crystal Reports XI and XI Release 2 include a new run-time licensing model. You can now install the reporting components on any number of servers within your organization without incurring additional licensing charges.

Distributing report files

There are three ways to distribute a report:

- Embed the report into an application's files.
- Maintain the report as separate files to be distributed with the application.
- Publish the report to a web service.

Embedded report files

When a report is added to a .NET Windows or web application, it is embedded into the application by default. An embedded report file is written directly into the application's binary source files; it is not loaded from a separate report file. The advantage is that no additional report files must be included when deploying the application and users cannot modify the report files. The disadvantage is that it tightly couples the report to the compiled application. If changes are made to the report, the entire application must be recompiled and redeployed.

NOTE	Embedded report files are not supported for web applications on the .NET 2.0 framework.

Non-embedded report (.rpt) files

A non-embedded report file is not compiled into an application's assembly. The report is distributed separately from the application and must be loaded separately. The report file is then independent of the application's .exe and libraries. The advantage of non-embedded reports is that modifying the report does not require that an application be recompiled and redeployed; only the reports are recompiled and redeployed. The disadvantage is that the number of files one must distribute and maintain increases and strongly-typed report objects cannot be used in the application.

To make a non-embedded report file, change the default setting for the property that defines how the report is added to the project.

To make a report file non-embeddable, complete these steps:

1. After the report is added to the project, choose the report in the Solution Explorer.
2. In the **Properties** dialog box, change the **Build Action** property from **Embedded Resource** to **None**.
3. Manually add the report file to the setup project. The report will not be compiled into the project's assembly and you will have to load it separately.

Web Services

When a report is generated as a web report, the report engine automatically creates an XML file that describes the report's public functions, input and output, and data types. Additionally, the engine publishes the file to a web server and makes the report data available as a web service. At run time, the report data is retrieved from the web service and displayed in the appropriate control. The report data can be consumed by any application capable of accessing a web service.

For example, a company creates a report showing its current unfilled orders. Exposed as a web service, vendors could reference this web service from their own applications and view pending orders in real time.

Deploying the Crystal Reports merge module

The .NET assemblies in Crystal Reports XI Release 2 have a single integrated merge module that must be included in your application. The merge module (crystal11_net_embeddedreporting.msm) is automatically installed to the C:\Program Files\Common Files\Merge Modules folder. If Crystal Reports is included in your application, the merge module is automatically detected when generating a deployment project. It includes all the dependent report components and assemblies needed to deploy your application. If map objects are used in the report files of your application, then the mapping merge module (CrystalReports11_maps.msm) must be added to the deployment project as well.

Installing the .NET Framework

You must ensure that the .NET framework is installed on the client computer. The .NET framework comes bundled with Visual Studio .NET and is also available from Microsoft.

Security-related changes

You may wish to migrate the security features already implemented in your RDC application. In addition to the security provided by these features, the .NET assemblies afford further security inherent in the .NET platform.

The .NET framework provides a broad range of security options that allows developers, administrators, and users to incorporate various

levels of security into their applications. These security features fall into three categories: authentication, authorization, and cryptography.

- Authentication is the verification of a user's identity. The .NET framework supports common authentication protocols, such as NTLM, Kerberos, and SSL client certificates. It allows developers to integrate .NET Passport authentication and to add cookie authentication into their applications.
- Authorization in .NET affords granular control to administrators, who can dictate what resources can be accessed by certain types of code. For example, code signed with a particular key having a specific hash value, etc.
- The .NET framework provides support for encryption, digital signatures, hash generation and decryption, and random number generation. Algorithms supported include DES, RSA, DSA, XML digital signature specification, and MD5 and SHA1 hashes.

The .NET assemblies include support for single sign-on (SSO), which allows you to incorporate reports into the security infrastructure. The new Update feature automatically provides you with the latest hot fixes and service packs to ensure new security vulnerabilities are addressed.

Licensing changes

The RDC and .NET components in Crystal Reports XI Release 2 make use of a concurrent processing license (CPL) model. Each component is set to three CPLs per process, meaning each component allows a maximum of three actions to be processed concurrently. Additional requests are queued until one of the three requests inside the engine completes.

As a general rule, if you anticipate your application regularly serving five or more concurrent users, consider using a server solution provided by Crystal Reports Server or BusinessObjects Enterprise. For high traffic applications, these server-based products offer more flexibility in terms of optimizing the deployment for performance.

More information on licensing is available at

http://www.businessobjects.com/products/reporting/crystalreports/licensing/details.asp.

Working with the object models in the .NET SDK

The .NET assemblies are available in the Visual Studio .NET IDE and the Developer Edition of Crystal Reports XI Release 2. They include an integrated design-time report authoring tool, a run-time report processing component, and an API to perform basic programmatic manipulation of report templates.

The object-oriented .NET technology has three main parts: the Common Language Runtime, the Framework classes, and ASP.NET. It hinges on the component-oriented programming paradigm. Software components often take the form of objects.

This section explains the object models in the .NET SDK. For a detailed comparison of code samples for the RDC and the .NET assemblies properties and methods, refer to Appendix A: Comparing RDC and .NET code.

Object models available in the .NET assemblies

The .NET assemblies provide an SDK that contains two object models:

- **CrystalReportViewer** is the simplest object model. The **CrystalReportViewer** control in the web or Windows form has, by definition, an underlying class of the same name. This **CrystalReportViewer** class exposes properties and methods for modifying the control's display functionality and for interacting with classes that manage database logon, parameters, and selection formulas. The **CrystalReportViewer** class exists in two different forms: one in a Windows namespace and one in a web namespace. Therefore, many elements of this object model are duplicated in both namespaces, with some variations to support the differences in the Windows and web platforms.

  The classes of this object model are contained within the CrystalDecisions.Web namespace for the web form version of this control or the CrystalDecisions.Windows.Forms namespace for the Windows form version of this control.

- **ReportDocument** is a more extensive object model. The **ReportDocument** class is a gateway to a set of classes in the Engine namespace, including **Database**, **DataDefinition**, **ExportOptions**, **PrintOptions**, **ReportDefinition**, **ReportOptions**, and **SummaryInfo**. This is an extensive set of classes that provide more powerful customization and interaction capability with the report.

  The classes of this object model are contained within the CrystalDecisions.CrystalReports.Engine namespace.

There have been many improvements in the licensing model for Crystal Reports. For more information, refer to this web page:

http://www.businessobjects.com/products/reporting/crystalreports/licensing/details.asp

CrystalReportViewer object model

The CrystalReportViewer class has two roles:

- **Report viewer controls** - In the web or Windows form, the **CrystalReportViewer** class is exposed as a .NET control that can be added to the form to display a report.

- **A simple object model** - In the underlying code-behind of the web or Windows form, the **CrystalReportViewer** class provides the simplest object model of the SDK.

The simplest way to place a report to your web or Windows form is to add the **CrystalReportViewer** control to the form, and then, in the code-behind class, assign the path of the report as a string to the **ReportSource** property of the **CrystalReportViewer** class.

By assigning the report path directly to the control rather than placing the report within a more complex object model, such as **ReportDocument** or **ReportClientDocument**, you restrict your interface to only the simplest object model, namely the **CrystalReportViewer**.

Report viewer controls

The .NET assemblies include standard form controls for report viewing. Each control encapsulates complex report layout information into a GUI object within the toolbox. Two controls are provided:

- **CrystalReportViewer** - This control displays the report in a page-based layout with the ability to move between pages.
- **CrystalReportPartsViewer** - This control displays report summary information in a small, portal-like window with the ability to drill down deeper into the report through a series of linked parts.

Each control is used to display a report by dropping the control from the toolbox onto the web or Windows form, and then assigning the **ReportSource** property of that control to a report.

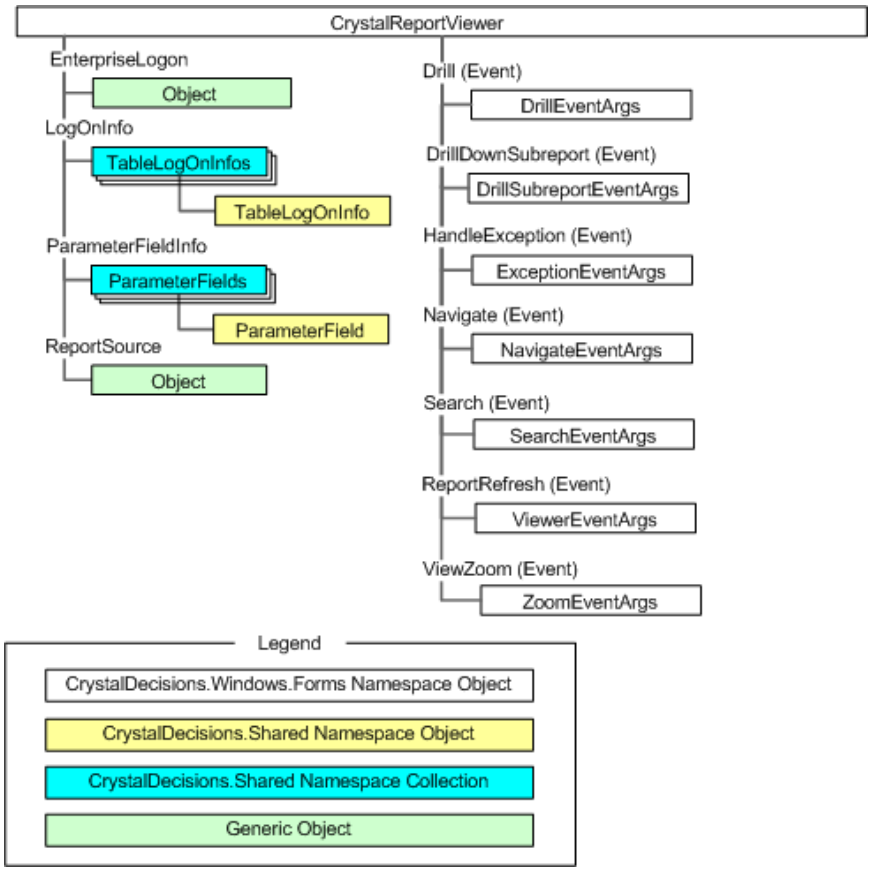
CrystalReportViewer object model diagrams

The diagrams below show the relationships within the CrystalReportViewer object model. **CrystalReportViewer** has two object model diagrams because the same control and class name (and therefore its underlying object model) is reused for both Windows and web applications. Therefore, the object model is repeated (with minor variations) in two different namespaces.

CrystalDecisions.Windows.Forms.CrystalReportViewer

The Windows version of the CrystalReportViewer object model (see Figure 1) contains the major classes and events that are shared by both the Windows and web versions of the object model.

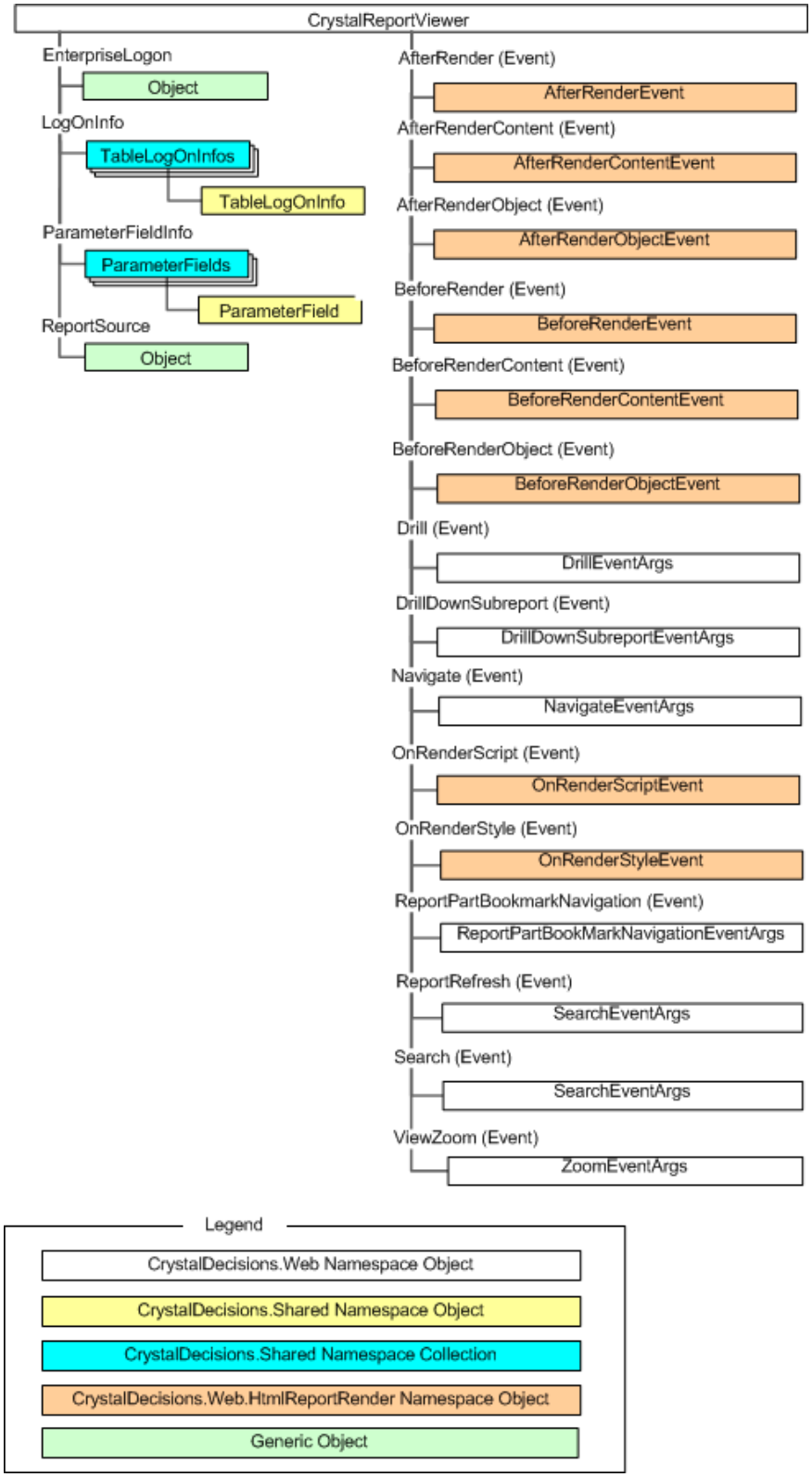
Figure 1 - Windows version of the CrystalReportViewer object model



CrystalDecisions.Web.CrystalReportViewer

The web version of the CrystalReportViewer object model (see Figure 2) contains the same classes and events as the Windows version, but in addition, it has events that are related to how ASPX pages load.

Figure 2 - Web version of the CrystalReportViewer object model



The ReportDocument object model

The **ReportDocument** class provides a more extensive object model for the SDK. The **ReportDocument** class is part of the CrystalDecisions.CrystalReports.Engine namespace. It functions as a gateway to a set of classes in the Engine namespace, which provides more tools for programmatic manipulation of a report.

The ReportDocument object model (see Figure 3) is used to encapsulate a report within an instance of a ReportDocument before it is assigned to the control. That provides access to the more complex and powerful object model provided in **ReportDocument.** Once you have introduced an additional object model, restrict the use of **CrystalReportViewer** code to only set display properties.

CAUTION	Because both object models are involved in this process (CrystalReportViewer to load and display the report, ReportDocument to encapsulate and manipulate the report), it is easy to confuse the roles of the two object models. Once the ReportDocument object model has been used to encapsulate the report, all report manipulation must be restricted to that model. If you try to use the properties and methods of the CrystalReportViewer object model to manipulate the report, the two object models will conflict and generate unexpected behavior.

Choosing the right object model

Before you build an application using the .NET assemblies, you must consider which object model to work with. To choose the correct object model for a project, you need to understand the best practices for working with the **CrystalReportViewer** control and know how the **CrystalReportViewer** interacts with the other object models.

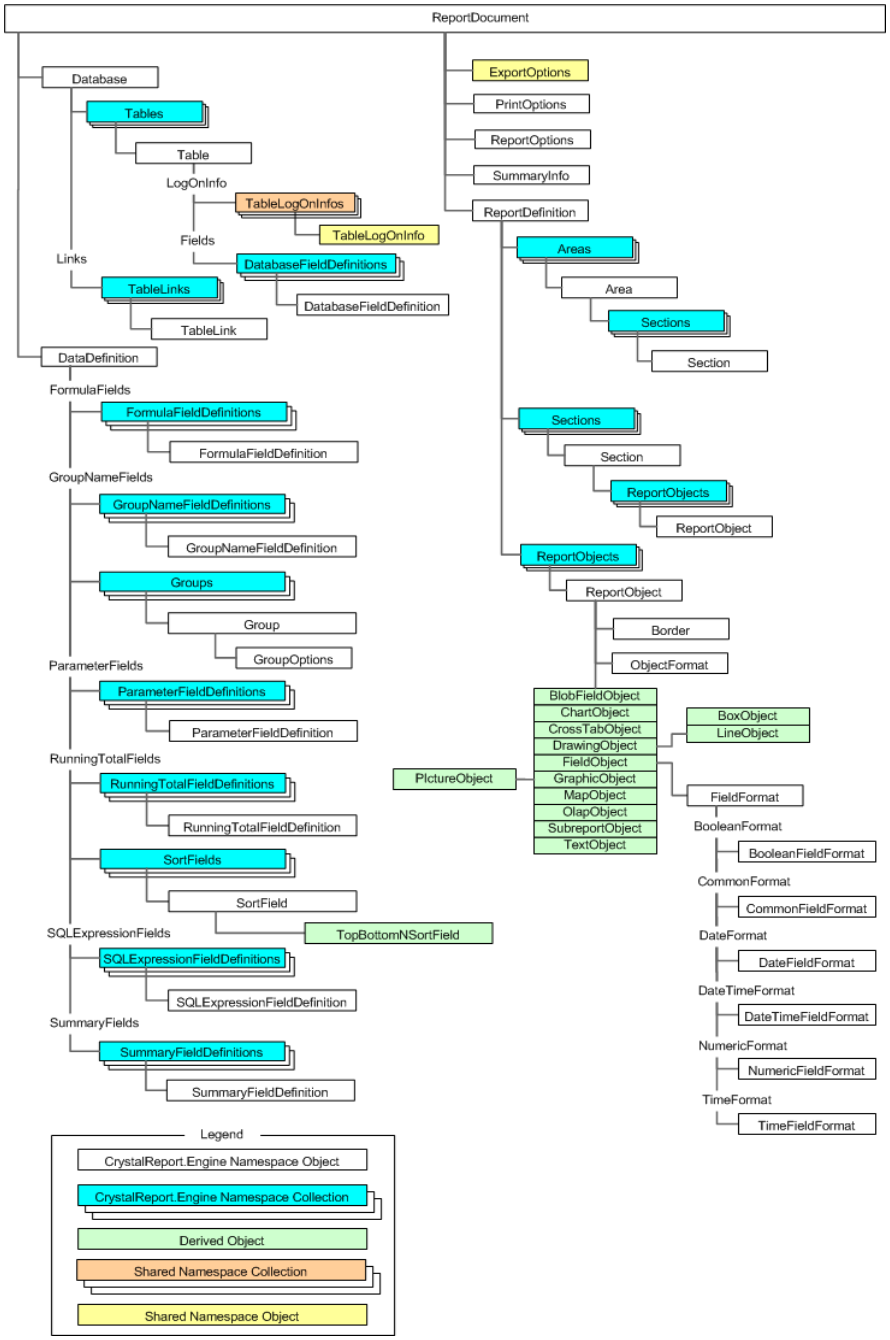
Best practices for working with the CrystalReportViewer control

An effective web application makes clear separation between its presentation layer and its underlying business logic. The design of the ASP.NET control naturally encourages this pattern. It encapsulates presentation information in the control and then binds that control to an underlying object or object model that performs the business logic.

The **CrystalReportViewer** is a .NET control that adheres to this same architecture. It functions as a display object on the web or Windows form (the presentation layer) and can be bound to any of the following report object models:

- ReportDocument object model
- Object models available by upgrade:
  - ReportClientDocument object model (Report Application Server)
  - InfoObject object model (Crystal Reports Server or BusinessObjects Enterprise)

Figure 3 - ReportDocument object model



Following best practices, the **CrystalReportViewer** control binds to one of these object models that performs the business logic, typically for report manipulation. In this scenario, the **CrystalReportViewer** control limits its programmatic interaction to modifying only display settings (for example, hiding or showing the viewer's toolbar or a button within that toolbar).

Choosing the correct object model

If you do not intend to upgrade to Crystal Reports Server or BusinessObjects Enterprise, the ReportDocument object model is the recommended object model for development work with the Crystal Reports SDK.

The **CrystalReportViewer** control contains properties and methods that enable interaction with how it displays reports. It also has a small number of properties and methods for interacting with reports that are bound to the control. These properties and methods constitute a limited object model.

Some report binding scenarios rely on the CrystalReportViewer object model. In these scenarios, the **CrystalReportViewer** control is bound directly to a report (for example, you pass in a path string to the report in a file directory) without first encapsulating the report in an object model. In these scenarios, because the **CrystalReportViewer** control encapsulates the report directly, you must rely on the control's limited object model for programmatic interaction with the report.

However, in most binding scenarios, using CrystalReportViewer as an object model is discouraged. Rather, use the ReportDocument object model for the following reasons:

- The CrystalReportViewer object model raises the risk of object model conflict.
- The CrystalReportViewer object model offers limited interaction with reports; it contains only a small subset of the features available in the ReportDocument object model.
- The CrystalReportViewer object model blurs the separation of the presentation layer and the underlying business logic in your code.

Do not mix the CrystalReportViewer object model with other object models. Use of the **CrystalReportViewer** control in its role as a limited object model works properly, provided you bind the control directly to the report with a simple path string.

If you encapsulate the report into one of the other object models (such as ReportDocument or ReportClientDocument) and then bind the control to that object model, stop using the CrystalReportViewer object model. The limited model that is provided by the **CrystalReportViewer** control becomes redundant to the more powerful object model to which it is bound. Also, settings that are applied to the CrystalReportViewer object model are visible to the other object model, which can result in unexpected behaviors and exceptions.

The rule of the thumb is to limit use of the **CrystalReportViewer** control to report display settings only when the control has been bound to one of the underlying object models.

Appendix A: Comparing RDC and .NET code

In this section, the code for a property or method of the RDC, followed by the .NET equivalent (provided here in VB.NET), are discussed.

Most RDC properties and methods have an equivalent property or method in .NET. Some RDC properties and methods are not implemented in .NET. Where this is the case, an alternative is suggested.

Report.PrintOut

RDC

Triggers printing of the report to the printer.

```
Report.Printout False
```

.NET

PrintToPrinter is a method of the ReportDocument object.

```
myReportDocument.PrintToPrinter(1, false, 0, 0)
```

RDC

To control the number of copies to print.

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

.NET

The first parameter of **PrintToPrinter** controls the number of copies to print.

```
myReportDocument.PrintToPrinter(3, true, 1, 5)
```

CRViewer.ViewReport

RDC

Viewing of the report is done through the Report Viewer control.

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

.NET

Viewing the report is done through the **CrystalReportViewer**.

```
myCrystalReportViewer.ReportSource = ReportDocument
myCrystalReportViewer.RefreshReport()
```

Report.Export

RDC

Export the report without prompting the user.

```
Report.Export False
```

.NET

The export related methods are methods of **ReportDocument**. However, there are multiple ways to export, depending on the format. For example, to export to disk you can use **Export** and also **ExportToDisk**.

Using Export:

```
Dim myExportOptions As New ExportOptions()
Dim myDiskFileDestinationOptions As _
  DiskFileDestinationOptions
myDiskFileDestinationOptions = _
  ExportOptions.CreateDiskFileDestinationOptions()
myExportOptions.ExportFormatType = _
  ExportFormatType.RichText
myExportOptions.ExportDestinationType = _
  ExportDestinationType.DiskFile
myDiskFileDestinationOptions.DiskFileName = fileName
myExportOptions.ExportDestinationOptions = _
  myDiskFileDestinationOptions
myReportDocument.Export(myExportOptions)
```

Using ExportToDisk:

```
Report.ExportToDisk(ExportFormatType.RichText, fileName)
```

DatabaseTable.ConnectionProperties

RDC

Using DatabaseTable's **ConnectionProperties**, you can specify the information to connect to the data source.

```
'Connect the first table of the tables collection to
'the data source
Dim dbProperties As CRAXDRT.ConnectionProperties
Set dbProperties = _
  Report.Database.Tables.Item(1).ConnectionProperties
dbProperties("DSN") = "Accounting"
dbProperties("Database") = "Administration"
```

```
dbProperties("User ID") = "734"
dbProperties("Password") = "bigboard"
```

.NET

ApplyLogOnInfo is a method of the **Table** object.

```
Dim myTableLogOnInfo As TableLogOnInfo
myTableLogOnInfo = New TableLogOnInfo _
  (myReportDocument.Database.Tables.Item(1).LogOnInfo)
Dim myConnectionInfo As New ConnectionInfo
myConnectionInfo.ServerName = "Accounting"
myConnectionInfo.DatabaseName = "Administration"
myConnectionInfo.UserID = "734"
myConnectionInfo.Password = "bigboard"
myTableLogOnInfo.ConnectionInfo = myConnectionInfo
myReportDocument.Database.Tables.Item(1).ApplyLogOnInfo _
  (myTableLogOnInfo)
```

RDC

Using **ConnectionProperties**, you can specify the database location.

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

.NET

Location is now a property on the **Table** object.

```
myReportDocument.Database.Tables.Item(1).Location = _
  "C:\new\xtreme.mdb"
```

Area.CopiesToPrint

RDC

Specifies the number of copies of each record in a section that should be printed.

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

.NET

Not supported in the .NET assemblies, but supported in the **ReportClientDocument** object model provided with the Report

Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

Report.SetDialogParentWindow

RDC

Specifies the handle of the parent window.

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

.NET

Not supported in the .NET assemblies, but supported in the **ReportClientDocument** object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

Report.DiscardSavedData

RDC

This method discards any data saved with the specified report.

```
Report.DiscardSavedData
```

.NET

The closest equivalent to discard saved data is to refresh the data in the report.

```
myReportDocument.Refresh()
```

ExportOptions.MailCcList

RDC

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

```
Report.ExportOptions.MailCcList = "John Brown; Jane Doe"
```

.NET

This is a property of the **MicrosoftMailDestinationOptions** object.

```
Dim myMicrosoftMailDestinationOptions As _
  MicrosoftMailDestinationOptions
myMicrosoftMailDestinationOptions = _
  ExportOptions.CreateMicrosoftMailDestinationOptions()
myMicrosoftMailDestinationOptions.MailCCList = _
  "John Brown; Jane Doe"
```

ExportOptions.MailMessage

RDC

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

```
Report.ExportOptions.MailMessage = "The meeting is at 4:00"
```

.NET

This is a property of the **MicrosoftMailDestinationOptions** object.

```
Dim myMicrosoftMailDestinationOptions As _
  MicrosoftMailDestinationOptions
myMicrosoftMailDestinationOptions = _
  ExportOptions.CreateMicrosoftMailDestinationOptions()
mailopts.MailMessage = "The meeting is at 4:00"
```

ExportOptions.MailSubject

RDC

Specifies the subject line in your e-mail message.

```
Report.ExportOptions.MailSubject = "Staff meeting"
```

.NET

This is a property of the **MicrosoftMailDestinationOptions** object.

```
Dim myMicrosoftMailDestinationOptions As _
  MicrosoftMailDestinationOptions
myMicrosoftMailDestinationOptions = _
  ExportOptions.CreateMicrosoftMailDestinationOptions()
myMicrosoftMailDestinationOptions.MailSubject = _
  "Staff meeting"
```

ExportOptions.MailToList

RDC

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

```
Report.ExportOptions.MailToList = "Jane Doe"
```

.NET

This is a property of the **MicrosoftMailDestinationOptions** object.

```
Dim myMicrosoftMailDestinationOptions As _
  MicrosoftMailDestinationOptions
myMicrosoftMailDestinationOptions = _
```

```
  ExportOptions.CreateMicrosoftMailDestinationOptions()
myMicrosoftMailDestinationOptions.MailToList = "Jane Doe"
```

ExportOptions.MailBccList

RDC

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

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

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.ExchangeFolderPath

RDC

Specifies the Exchange path, when you want to export to Microsoft Exchange.

```
Report.ExportOptions.ExchangeFolderPath = _
  "C:\Microsoft\Exchange\NewRpt.rpt"
```

.NET

This is a property of the **ExchangeFolderDestinationOptions** object.

```
Dim myExchangeFolderDestinationOptions As _
  ExchangeFolderDestinationOptions = _
  ExportOptions.CreateExchangeFolderDestinationOptions()
myExchangeFolderDestinationOptions.FolderPath = _
  "C:\Microsoft\Exchange\NewRpt.rpt"
```

ExportOptions.ExchangeProfile

RDC

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

```
Report.ExportOptions.ExchangeProfile = "James Andrews"
```

.NET

This is a property of the **ExchangeFolderDestinationOptions** object.

```
Dim myExchangeFolderDestinationOptions As _
```

```
  ExchangeFolderDestinationOptions = _
  ExportOptions.CreateExchangeFolderDestinationOptions()
myExchangeFolderDestinationOptions.Profile = _
  "James Andrews"
```

ExportOptions.ExchangePassword

RDC

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

```
Report.ExportOptions.ExchangePassword = "Pickle"
```

.NET

This is a property of the **ExchangeFolderDestinationOptions** object.

```
Dim myExchangeFolderDestinationOptions As _
  ExchangeFolderDestinationOptions = _
  ExportOptions.CreateExchangeFolderDestinationOptions()
myExchangeFolderDestinationOptions.Password = "Pickle"
```

FormulaFieldDefinition.Text

RDC

Specifies a new string for an existing formula.

```
'Pass the formula to the first formula in the
'FormulaFields collection
Report.FormulaFields.Item(1).Text = "{file.SALES} *.1"
'Pass the formula to the second formula in the
'FormulaFields collection
Report.FormulaFields.Item(2).Text = _
  "{file.SALES} + {file.COMMISSION}"
```

.NET

This is a property of the **FormulaFieldDefinition** object.

```
'Pass the formula to the first formula in the
'FormulaFields collection
Report.FormulaFields.Item(0).Text = "{file.SALES} *.1"
'Pass the formula to the second formula in the
'FormulaFields collection
Report.FormulaFields.Item(1).Text = _
  "{file.SALES} + {file.COMMISSION}"
```

Report.GroupSelectionFormula

RDC

Specifies the groups to be used when printing the report.

```
Report.GroupSelectionFormula = _
  "Sum ({order details.ORDER AMOUNT}, _
        {customer.CUSTOMER ID}) < $10000"
```

.NET

This is a property of the **DataDefinition** object.

```
myReportDocument.DataDefinition.GroupSelectionFormula = _
  "Sum({order details.ORDER AMOUNT}, _
       {customer.CUSTOMER ID}) < $10000"
```

GroupSortFields.Add

RDC

This is a property of the **Report** object's **GroupSortFields** collection.

This scenario requires a report that already contains a Group that must also contain a Summary field. A Group Sort field can only exist if the group contains a Summary field because the sort is based on the Summary field. In this example, the report is grouped on {Customer.Region} and the summary field is the "SUM of Customer.Last Year's Sales (Currency)".

```
'Declare a SummaryFieldDefinition Object
Dim crxSummaryField As CRAXDRT.SummaryFieldDefinition
'Getting the first Summary Field which is the '"SUM of
'Customer.Last Year's Sales"
Set crxSummaryField = Report.SummaryFields.Item(1)
'Add the Group Sort Field
Report.GroupSortFields.Add crxSummaryField, _
  crDescendingOrder
```

.NET

```
Dim mySummaryFieldDefinition As SummaryFieldDefinition
mySummaryFieldDefinition = _
  yReportDocument.DataDefinition.SummaryFields.Item(1)
Dim mySortField As SortField = _
  myReportDocument.DataDefinition.SortFields.Item(1)
mySortField.Field = mySummaryFieldDefinition
mySortField.SortDirection = SortDirection.DescendingOrder
```

Report.BottomMargin

RDC

Sets the bottom margin for the specified report.

```
Report.BottomMargin = 720
```

.NET

This is a property of the **PageMargins** object.

```
myReportDocument.PrintOptions.PageMargins.bottomMargin = _
  720
```

Report.LeftMargin

RDC

Sets the left margin for the specified report.

```
Report.LeftMargin = 1440
```

.NET

This is a property of the **PageMargins** object.

```
myReportDocument.PrintOptions.PageMargins.leftMargin = 1440
```

Report.RightMargin

RDC

Sets the right margin for the specified report.

```
Report.RightMargin = 1440
```

.NET

This is a property of the **PageMargins** object.

```
myReportDocument.PrintOptions.PageMargins.rightMargin = _
  1440
```

Report.TopMargin

RDC

Sets the top margin for the specified report.

```
Report.TopMargin = 720
```

.NET

This is a property of the **PageMargins** object.

```
myReportDocument.PrintOptions.PageMargins.topMargin = 720
```

ParameterFieldDefinition.AddCurrentValue

RDC

Changes the default value of the specified parameter field.

```
'Note the RDC has the ability to add multiple values to
'a single parameter.
'Add the value to the first parameter in the
'ParameterFields collection
Report.ParameterFields.Item(1).AddCurrentValue 1000
Report.ParameterFields.Item(1).AddCurrentValue 5000
Report.ParameterFields.Item(1).AddCurrentValue 10000
```

.NET

```
myReportDocument.ParameterFields(0).CurrentValues _
  .AddValue(1000)
myReportDocument.ParameterFields(0).CurrentValues _
  .AddValue(5000)
myReportDocument.ParameterFields(0).CurrentValues _
  .AddValue(10000)
```

Report.PrintDate

RDC

Sets the entire PrintDate.

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

.NET

Not supported.

Report.SelectPrinter

RDC

The first parameter sets the name of the printer driver that is to print the report. The second parameter sets the name of the printer that is to print the report. The third parameter sets the name of the printer port for the specified printer:

```
object.SelectPrinter DriverName, PrinterName, PortName
Report.SelectPrinter ""Epson24.drv"", " Epson LQ-850", _
  "LPT1"
```

.NET

Does not have properties to set the printer driver name or port, but does have a property to set the printer name.

```
myReportDocument.PrintOptions.PrinterName = "Epson LQ-850"
```

ExportOptions.CharStringDelimiter

RDC

Sets the character to separate strings.

```
Report.ExportOptions.CharStringDelimiter = ","
```

.NET

```
Dim myCharacterSeparatedValuesFormatOptions As _
  CharacterSeparatedValuesFormatOptions
myCharacterSeparatedValuesFormatOptions = ExportOptions _
  .CreateCharacterSeparatedValuesFormatOptions()
myCharacterSeparatedValuesFormatOptions.Delimiter = ","
```

ExportOptions.CharFieldDelimiter

RDC

Sets the character to separate fields.

```
Report.ExportOptions.CharFieldDelimiter = "@"
```

.NET

```
Dim myCharacterSeparatedValuesFormatOptions As _
  CharacterSeparatedValuesFormatOptions
myCharacterSeparatedValuesFormatOptions = ExportOptions _
  .CreateCharacterSeparatedValuesFormatOptions()
myCharacterSeparatedValuesFormatOptions.SeparatorText = "@"
```

ExportOptions.NumberOfLinesPerPage

RDC

Sets the number of lines per page.

```
Report.ExportOptions.NumberOfLinesPerPage = 50
```

.NET

This is a property of the **TextFormatOptions** object.

```
Dim myTextFormatOptions As New TextFormatOptions
myTextFormatOptions.LinesPerPage = 60
```

ExportOptions.DiskFileName

RDC

Sets the path and name of the exported file.

```
Report.ExportOptions.DiskFileName = "C:\crw\cust_rpt.txt"
```

.NET

```
Dim myDiskFileDestinationOptions As _
  New DiskFileDestinationOptions
myDiskFileDestinationOptions.DiskFileName = _
  "C:\crw\cust_rpt.txt"
```

ExportOptions.ODBCDataSourcePassword

RDC

Sets the password to the ODBC data source.

```
Report.ExportOptions.ODBCDataSourcePassword = "merry%%5"
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.ODBCDataSourceName

RDC

Sets the ODBC data source name.

```
'Set the ODBC data source name
Report.ExportOptions.ODBCDataSourceName = "pickle"
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.ODBCExportTableName

RDC

Sets the ODBC data source table name.

```
Report.ExportOptions.ODBCExportTableName = "Employees"
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.ODBCDataSourceUserID

RDC

Set the user ID for the ODBC data source.

```
Report.ExportOptions.ODBCDataSourceUserID = "LisaB"
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.FormatType

RDC

Set the format type to tab separated values.

```
Report.ExportOptions.FormatType = crEFTTabSeparatedValues
```

.NET

```
Dim myExportOptions = new ExportOptions
myExportOptions.ExportFormatType = _
  ExportFormatType.TabSeperatedText
```

ExportOptions.UseReportDateFormat

RDC

Set the date format to be the same as report.

```
Report.ExportOptions.UseReportDateFormat = True
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ExportOptions.UseReportNumberFormat

RDC

Set the number format to be the same as report

```
Report.ExportOptions.UseReportNumberFormat = True
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

Report.DisplayProgressDialog

RDC

Enables or disables the display of the progress dialog box. The progress dialog box displays the progress of the report when it is running.

```
Report.DisplayProgressDialog = False
```

.NET

Not supported.

PrintingStatus.NumberOfRecordPrinted

RDC

Determines the number of records actually printed.

```
'Preview the Report
CRViewer1.ReportSource = Report
CRViewer1.ViewReport
'Pass the number of records printed to the Printed variable
Printed& = Report.PrintingStatus.NumberOfRecordPrinted
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). The **RowsetController** allows access to the rowset of the report, including the number of records fetched. RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

PrintingStatus.NumberOfRecordRead

RDC

Determines the number of records actually processed.

```
'Read the records into the report
Report.ReadRecords
'Pass the number of records read to the Read variable
Read& = Report.PrintingStatus.NumberOfRecordRead
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). The **RowsetController** allows access to the rowset of the report, including the number of records fetched. RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

PrintingStatus.NumberOfRecordSelected

RDC

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

```
'Read the records into the report
Report.ReadRecords
'Pass the number of records selected to the
'Selected variable
Selected& = Report.PrintingStatus.NumberOfRecordSelected
```

.NET

Not supported.

CRViewer.GetCurrentPageNumber

RDC

Gets the current page number

```
Result% = CRViewer1.GetCurrentPageNumber
```

.NET

```
Dim result As Integer = _
  CrystalReportViewer1.GetCurrentPageNumber()
```

Application.OpenReport

RDC

This is a method of the **Application** object used for opening reports saved in the Crystal Reports format (.rpt). The RDC can also open reports saved as ActiveX Designers (.dsr) within VB.

Opening a report:

```
'General Declarations
'Declare an application object
Dim crxApplication as New craxdrt.Application
'Declare a Report object
Dim Report as craxdrt.Report
'In a function or Sub procedure
Set Report = _
  crxApplication.OpenReport("C:\crw\company.rpt", 1)
```

Declaring a variable as a new DSR:

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

Setting a **Report** object to a DSR:

```
'General Declarations
Dim Report as craxdrt.Report
Private Sub Form_Load()
  'Set the generic Report object to the DSR
  Set Report = New CrystalReport1
End Sub
```

.NET:

ActiveX Designers (.dsr) are not supported in the .NET assemblies, but is supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

Opening a report:

```
myReportDocument.Load(fileName)
```

Report.ReportTitle

RDC

Sets the title of the report.

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

.NET

```
myReportDocument.ReportTitle = "My Report"
```

ReportObject.Font

RDC

Access and set the font properties for the specified report object.

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

```
'Declare a Section object
Dim crxSection As CRAXDRT.Section
'Declare a generic object
Dim crxObject As Object
'Declare a Field object
Dim crxFieldObject As CRAXDRT.FieldObject
'Set the Font properties of the first field in the
'Report Objects collection of the Detail Section
'Set font name
```

```
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Name = "Arial"
'Set font size
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Size = 10
'Set font italic
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Italic = True
'Set font bold
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Bold = True
'Set font underline
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Underline = True
'Set font strikethrough
Report.Sections.Item("D").ReportObjects.Item(1) _
  .Font.Strikethrough = False
```

.NET

ApplyFont is a method of the **TextObject** object.

```
Dim myTextObject As TextObject
Dim myReportObject As ReportObject = _
  myReportDocument.ReportDefinition.Sections.Item("D"). _
    ReportObjects.Item(1)
If TypeOf (myReportObject) Is TextObject Then
  myTextObject = myReportObject
  Dim font As New Font("Arial", 10, _
    FontStyle.Italic And _
    FontStyle.Bold And _
    FontStyle.Underline And FontStyle.Strikeout)
  myTextObject.ApplyFont(font)
End If
```

Section

RDC

Access and set properties of a section. The **Sections** collection is a property of the Report.

If the report is saved in a Crystal Reports format (.rpt), then the properties can be set through the **Section** object. If the report is saved in an ActiveX Designer format (.dsr), then the properties can be set through

the section's property window, in the Section Format event, or through the **Section** object.

```
'Set the Detail section's Suppress property
Report.Sections.Item("D").Suppress = False
'Set the section's NewPageBefore property
Report.Sections.Item("D").NewPageBefore = False
'Set the section's NewPageAfter property
Report.Sections.Item("D").NewPageAfter = False
'Set the section's KeepTogether property
Report.Sections.Item("D").KeepTogether = True
'Set the section's SuppressIfBlank property
Report.Sections.Item("D").SuppressIfBlank = True
'Set the section's ResetPageNumberAfter property
Report.Sections.Item("D").ResetPageNumberAfter = False
'Set the section's PrintAtBottomOfPage property
Report.Sections.Item("D").PrintAtBottomOfPage = False
'Set the section's UnderlaySection property
Report.Sections.Item("D").UnderlaySection = True
'Set the section's BackColor property
Report.Sections.Item("D").BackColor = vbRed
```

.NET

Section collection is a property of the **ReportDefinition** object.

```
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableSuppress = False
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableNewPageBefore = False
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableNewPageAfter = False
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableKeepTogether = True
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableSuppressIfBlank = True
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableResetPageNumberAfter = False
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnablePrintAtBottomOfPage = False
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.EnableUnderlaySection = True
```

```
myReportDocument.ReportDefinition.Sections.Item(1). _
  SectionFormat.BackgroundColor = Color.Red
```

Section.Height

RDC

Sets the height of the section.

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

.NET

```
myReportDocument.ReportDefinition.Sections.Item(1)
```

Report.RecordSelectionFormula

RDC

Specifies the records to be used with the report.

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

.NET

```
myReportDocument.RecordSelectionFormula  = "{file.QTY} > 5"
```

DatabaseTable.SetSessionInfo

RDC

Use **SetSessionInfo** method of the **DatabaseTable** object.

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

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

SortField.Field, SortField.SortDirection

RDC

These are properties of the **SortField** object. **Field** sets the field to sort on, while **SortDirection** sets the sort direction.

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

```
'Currently the sort is based on the Customer Name field and
'the application is to change it to the 'Last Year's
'Sale's' field. This field must be present on the report.
'Accessing the first table to get the 8th field
Set crxDatabaseField = _
  Report.Database.Tables.Item(1).Fields.Item(8)
'Set the field to crxDatabaseField
Report.RecordSortFields.Item(1).Field = crxDatabaseField
'Set the SortField direction
Report.RecordSortFields.Item(1).SortDirection = _
  crAscendingOrder
```

.NET

```
Dim myDatabaseFieldDefinition As DatabaseFieldDefinition
myDatabaseFieldDefinition = _
  myReportDocument.Database.Tables.Item(1).Fields.Item(8)

myReportDocument.DataDefinition.SortFields.Item(1) _
  .Field = _myDatabaseFieldDefinition

myReportDocument.DataDefinition.SortFields.Item(1) _
  .SortDirection = _SortDirection.AscendingOrder
```

PrintingStatus.Progress

RDC

Determines the status of printing.

```
'Print the Report
Report.Printout
'Pass the Progress of the Print job to the Status variable
Status & = Report.PrintingStatus.Progress
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ParameterFieldDefinition.AddCurrentValue

RDC

This is a property of the **ParameterFieldDefinition** object. All parameters including stored procedures and Crystal parameters are set through the **ParameterFieldDefinition** object.

```
'Set the value of the Stored procedure which is the
'first parameter in the ParameterFields collection
Report.ParameterFields.Item(1).AddCurrentValue = _
  "06/14/1989"
```

.NET

```
myReportDocument.ParameterFields(1).CurrentValues _
  .AddValue("06/14/1989")
```

Report.OpenSubreport

RDC

This is a method of the **SubreportObject** object and the **Report** object. Pass the name of the subreport to the **OpenSubreport** method of the **Report** object. This method is valid for reports saved in the Crystal Report (.rpt) format or as an ActiveX Designer (.dsr) in VB.

```
'Declare Report object to set to the Subreport
Dim crxSubreport As CRAXDRT.Report
'Set crxSubreport to the subreport "sub1"
Set crxSubreport = Report.OpenSubreport("sub1")
```

.NET

```
Dim subreport As ReportDocument
subreport = myReportDocument.OpenSubreport("sub1")
```

SubreportObject.OpenSubreport

RDC

Open the subreport by setting a subreport to a valid **SubreportObject** in the report. Searching through the sections and report objects in the section accesses the subreport. Valid for .rpt and .dsr formats.

```
'Declare Report object to set to the Subreport
Dim crxSubreport As CRAXDRT.Report
'Declare a Section object
Dim crxSection As CRAXDRT.Section
'Declare a Generic object
Dim crxObject As Object
'Declare a SubreportObject Object
```

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

.NET

```
Dim myReportDocument2 As ReportDocument
Dim mySection As Section
Dim myReportObject As ReportObject
Dim mySubreportObject As SubreportObject
For Each mySection In _
    myReportDocument.ReportDefinition.Sections
  For Each myReportObject In mySection.ReportObjects
    If myReportObject.Kind = _
        ReportObjectKind.SubreportObject Then
      mySubreportObject = myReportObject
      myReportDocument2 = mySubreportObject.OpenSubreport _
        (mySubreportObject.SubreportName)
    End If
  Next
Next
```

RDC

If the location of the subreport is known, the number of lines of code can be reduced. For example, the subreport is the first report object in the first section (Report Header) of the report. Valid for .rpt and .dsr formats.

```
'Declare Report object to set to the Subreport
Dim crxSubreport As CRAXDRT.Report
'Set crxSubreport to the first object in the Report Header
```

```
'section.
Set crxSubreport = _
  Report.Sections.Item("RH").ReportObjects.Item(1) _
    .OpenSubreport
```

.NET

```
Dim subreportDocument As ReportDocument
Dim mySubreportObject as SubreportObject
mySubreportObject = _
  myReportDocument.ReportDefinition.Sections.Item("RH") _
    .ReportObjects.Item(1)
subreportDocument = _
  mySubreportObject.OpenSubreport _
    (mySubreportObject.SubreportName)
```

CRViewer.EnableDrillDown

RDC

Indicates whether drill-down on summary values is allowed in the report viewer control.

```
CRViewer1.EnableDrillDown = True
```

.NET

```
myCrystalReportViewer.EnableDrillDown = True
```

CRViewer.EnableToolbar

RDC

Specifies whether the toolbar is to appear in the Report Viewer control.

```
CRViewer1.EnableToolbar = True
```

.NET

```
myCrystalReportViewer.DisplayToolbar = True;
```

CRViewer.Height

RDC

Sets the height of the Report Viewer control within the parent form.

```
'To set the height to match the parent form
'Place within the Form Resize event.
CRViewer1.Height = ScaleHeight
```

.NET

```
myCrystalReportViewer.Height = ScaleHeight
```

CRViewer.Left

RDC

Sets the left side of the Report Viewer control within the parent form.

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

.NET

```
myCrystalReportViewer.Left = 0
```

CRViewer.EnableStopButton

RDC

Specifies whether a stop button is available in the Report Viewer control.

```
CRViewer1.EnableStopButton = True
```

.NET

```
myCrystalReportViewer.ShowCloseButton = True
```

CRViewer.EnableCloseButton

RDC

Specifies whether a close button is available in the Report Viewer control.

```
CRViewer1.EnableCloseButton = True
```

.NET

```
myCrystalReportViewer.ShowCloseButton = True
```

CRViewer.EnableExportButton

RDC

Specifies whether an export button is available in the Report Viewer control.

```
CRViewer1.EnableExportButton = True
```

.NET

```
myCrystalReportViewer.ShowExportButton = True
```

CRViewer.EnableGroupTree

RDC

Specifies whether a group tree appears in the Report Viewer control.

```
CRViewer1.EnableGroupTree = True
```

.NET

```
myCrystalReportViewer.DisplayGroupTree = True
```

CRViewer.EnableNavigationControls

RDC

Specifies whether the navigation controls appear in the Report Viewer control.

```
CRViewer1.EnableNavigationControls = True
```

.NET

```
myCrystalReportViewer1.ShowPageNavigateButtons = True
```

CRViewer.EnablePrintButton

RDC

Specifies whether a print button appears in the Report Viewer control.

```
CRViewer1.EnablePrintButton = True
```

.NET

```
myCrystalReportViewer1.ShowPrintButton = true;
```

CRViewer.EnableProgressControl

RDC

Specifies whether the progress control appears in the Report Viewer control.

```
CRViewer1.EnableProgressControl = True
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

CRViewer.EnableRefreshButton

RDC

Specifies whether a refresh button appears in the Report Viewer control.

```
CRViewer1.EnableRefreshButton = True
```

.NET

```
myCrystalReportViewer.ShowRefreshButton = True
```

CRViewer.EnableSearchControl

RDC

Specifies whether a search control appears in the Report Viewer control.

```
CRViewer1.EnableSearchControl = True
```

.NET

```
myCrystalReportViewer.ShowTextSearchButton = True
```

CRViewer.EnableZoomControl

RDC

Specifies whether a zoom control appears in the Report Viewer control.

```
CRViewer1.EnableZoomControl = True
```

.NET

```
myCrystalReportViewer.ShowZoomButton = True
```

CRViewer.Top

RDC

This is a property of the Report Viewer control and sets the top of the control within the parent form.

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

.NET

```
myCrystalReportViewer.Top = 0
```

CRViewer.Width

RDC

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

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

.NET

```
myCrystalReportViewer.Width = width
```

Report.RecordSelectionFormula

RDC

This is a read and write property of the **Report** object.

```
SelectionFormula$ = Report.RecordSelectionFormula
```

.NET

```
Dim selectionFormula as String = _
  myReportDocument.RecordSelectionFormula
```

ReportObject.SubreportName

RDC

Name of the subreport.

```
'Declare a variable to hold the name of the subreport
Dim strSubreport as String
'The subreport is the first Report object in the
'Report Footer section
strSubreport = _
  Report.Sections.Item("RF").ReportObjects.Item(1) _
    .SubreportName
```

.NET

```
Dim mySubreportObject As SubreportObject
mySubreportObject = _
  myReportDocument.ReportDefinition.Sections.Item("RF") _
    .ReportObjects.Item(1)
strSubreport = mySubreportObject.SubreportName
```

Application.LogonServer, DataBase.LogonServer

RDC

Logon to the specified server. This is a method of the **Application** and **Database** Objects.

Application object:

```
CrxApplication.LogonServer "pdsodbc.dll", "Accounting", _
  "Administration", "bobg", "bigboard"
```

Database object:

```
'Log onto the data source
Report.DataBase.LogonServer "pdsodbc.dll", "Accounting", _
  "Administration", "bobg", "bigboard"
```

.NET

```
Dim myTableLogOnInfo As TableLogOnInfo
myTableLogOnInfo = New TableLogOnInfo _
  (myReportDocument.Database.Tables.Item(1).LogOnInfo)

Dim myConnectionInfo As New ConnectionInfo

myConnectionInfo.ServerName = "Accounting"
myConnectionInfo.DatabaseName = "Administration"
myConnectionInfo.UserID = "bobg"
myConnectionInfo.Password = "bigboard"
myTableLogOnInfo.ConnectionInfo = myConnectionInfo

myReportDocument.Database.Tables.Item(1) _
  .ApplyLogOnInfo(myTableLogOnInfo)
```

CRViewer.ShowFirstPage

RDC

Displays the first page of the report in the Report Viewer control.

```
CRViewer1.ShowFirstPage
```

.NET

```
myCrystalReportViewer.ShowFirstPage()
```

CRViewer.ShowLastPage

RDC

Displays the last page of the report in the Report Viewer control.

```
CRViewer1.ShowLastPage
```

.NET

```
myCrystalReportViewer.ShowLastPage()
```

CRViewer.ShowNextPage

RDC

Displays the next page of the report in the Report Viewer control.

```
CRViewer1.ShowNextPage
```

.NET

```
myCrystalReportViewer.ShowNextPage()
```

CRViewer.ShowPreviousPage

RDC

Displays the previous page of the report in the Report Viewer control.

```
CRViewer1.ShowPreviousPage
```

.NET

```
myCrystalReportViewer.ShowPreviousPage()
```

CRViewer.ShowNthPage

RDC

Displays the nth page of the report in the Report Viewer control.

```
CRViewer1.ShowNthPage 3
```

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

```
'Set the Report to the Report Viewer
CRViewer1.ReportSource = Report
'View the Report
CRViewer1.ViewReport
'Continue loop while the Report Viewer is busy
'downloading data
While CRViewer1.IsBusy
  DoEvents
Wend
'Set the report to preview on the third page
CRViewer1.ShowNthPage 3
```

.NET

```
CrystalReportViewer1.ShowNthPage(3)
```

CRViewer.Zoom

RDC

Sets the magnification factor for the report in the Report Viewer control.

```
CRViewer1.Zoom 150
```

.NET

```
CrystalReportViewer1.Zoom(zoomLevel)
```

Reset Report

RDC

To reset the report, set the **Report** object to "Nothing," then open the report again and set the new properties.

```
'Set the Report object to Nothing
Report = Nothing
```

.NET

```
myReportDocument = nothing
```

DatabaseTable.Location

RDC

Sets the location of the database.

```
'Get the number of tables in the report
'from the count property of the DatabaseTables collection
nTables% = Report.Database.Tables.Count
'Set the location the first DatabaseTable in the
'DatabaseTables collection
Report.Database.Tables.Item(1).Location = _
  "C:\new\xtreme.mdb"
```

.NET

```
int nTables = myReportDocument.Database.Tables.Count
myReportDocument.Database.Tables.Item(1).Location = _
  "C:\new\xtreme.mdb"
```

DatabaseTable.SetLogOnInfo

RDC

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

```
Report.Database.Tables.Item(1).SetLogOnInfo "DSN", _
  "Database", "User ID", "Password"
```

.NET

```
Dim myTableLogOnInfo As TableLogOnInfo
myTableLogOnInfo = New TableLogOnInfo _
```

```
  (myReportDocument.Database.Tables.Item(1).LogOnInfo)

Dim myConnectionInfo As New ConnectionInfo

myConnectionInfo.ServerName = "DSN"
myConnectionInfo.DatabaseName = "Database"
myConnectionInfo.UserID = "User ID"
myConnectionInfo.Password = "Password"
myTableLogOnInfo.ConnectionInfo = myConnectionInfo
myReportDocument.Database.Tables.Item(1) _
  .ApplyLogOnInfo(myTableLogOnInfo)
```

Report.SQLQueryString

RDC

This is a property of the **Report** object.

```
'Declare a string variable to hold the SQL Query
Dim strSQL As String
'To retrieve the SQL Query
strSQL = Report.SQLQueryString
'To pass a SQL Query
Report.SQLQueryString = strSQL
```

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

ParameterFieldDefinition

RDC

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

```
'Retrieve the first default value in the parameter
'RDC has the ability to add multiple values
nParameterValue = _
  Report.Parameters.Item(1).GetNthDefaultValue(1)
'Set the first parameter in the ParameterFields collection
Report.Parameters.Item(1).AddCurrentValue nParameterValue
'Get the number of parameters in the report
```

```
'from the count property of the ParameterFields collection
nParameters% = Report.ParameterFields.Count
```

.NET

```
Dim myParameterValue As ParameterValue
myParameterValue = myReportDocument _
  .ParameterFields.Item(1).DefaultValues.Item(1)
myReportDocument.ParameterFields.Item(1) _
  .CurrentValues.AddValue(myParameterValue)
Dim nParameters As Integer = _
  myReportDocument.ParameterFields.Count
```

DatabaseTable.SetPrivateData

RDC

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

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

.NET

Not supported in the .NET assemblies, but supported in the ReportClientDocument object model provided with the Report Application Server (RAS). RAS is available with Crystal Reports Server and BusinessObjects Enterprise.

Appendix B: Feature comparison of RDC and .NET assemblies

In this section, two tables are presented:

- A feature table that compares features supported in RDC and the .NET assemblies.
- A functional matrix that compares functions supported in RDC and the .NET assemblies.

These tables focus only on the change in support between the two components.

** [] denotes indirectly through .NET platform

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
Parameters			
			Unchanged
Database			
Field Mapping	Y		
Show SQL Query	Y		
Perform Grouping on Server	Y		
Runtime Dataset support (ADO.NET)		Y	
Runtime Dataset support (ADO)	Y		
Runtime Dataset support (XML)		[]	Through Visual Studio .NET data providers
Exporting			
From Subreports			
Destinations:			
Application	Y		
Lotus domino	Y		
Lotus Domino mail	Y		
HTTP Response		Y	
Stream		Y	
Formats:			
Tab separated text	Y		
XML	Y		
Report Definition	Y		
ODBC	Y		
Editable RTF	Y		

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
Printing			
			Unchanged
Report Object Common Properties			
ConditionFormula	RW		
HyperlinkType	RW		
HyperlinkText	RW		
LinkedURI	RW		
BlobField Object			
Insert new	Y		
Box Object			
Insert new	Y		
Properties:			
BottomRightSection	RW	R	
CornerEllipseHeight	RW		
CornerEllipseWidth	RW		
Chart Object			
Properties:			
Chart type:	RW		
Bar	Y		
Line	Y		
Area	Y		
Pie	Y		
Doughnut	Y		
3D Riser	Y		
3D Surface	Y		
XY Scatter	Y		
Radar	Y		
Bubble	Y		
Stock	Y		
Data type:			
Group / Cross-tab	Y		
Title	RW		
Subtitle	RW		
Footnote	RW		
TitleFont	RW		
SubtitleFont	RW		
FootnoteFont	RW		
ConditionFields	RW		
DataFields	RW		

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
Other properties	RW		E.g.: GroupsTitle, SeriesTitle, XAxisTitle, MaxDataAxisValue, GroupAxisGridline, DataPoint, SliceDetachment, etc.
Crosstab Object			
Insert new	Y		
Properties:			
LineThickness	RW		
ShowCellMargins	RW		
SuppressEmptyRows	RW		
SuppressEmptyColumns	RW		
KeepColumnsTogether	RW		
SuppressRowGrandTotals	RW		
SuppressColumnGrandTotals	RW		
RowGrandTotalColor	RW		
ColumnGrandTotalColor	RW		
EnableRepeatRowLabels	RW		
SummaryFields	RW		
RowGroups	RW		
ColumnGroups	RW		
Field Object			
Insert new	Y		
Line Object			
Insert new	Y		
Properties:			
BottomRightSection	RW	R	
Map Object			
			Unchanged
OLAPGrid Object			
			Unchanged
OLE Object			
Insert new	Y		
Properties:			

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
LeftCropping	RW		
TopCropping	RW		
RightCropping	RW		
BottomCropping	RW		
XScaling	RW		
YScaling	RW		
Other properties	RW		FormattedPicture, GetLinkSource
Subreport Object			
Insert new	Y		
Properties:			
SubreportName	RW	R	
SubreportLinks	RW		
ReimportSubreport	RW		
Text Object			
Insert new	Y		
Properties:			
MaxNumberOfLines	RW		
EnableSuppressIfDuplicated	RW		
FieldElements	RW		
FieldHeading Object			Derived from Text object
Modify FieldHeading object Properties:			
FieldObjectName		R	
Report Options			
PushDownGroupBy	Y		
SelectDistinctRecords	Y		
AsyncQuery	Y		
CanSelectDistinctRecords	Y		
SaveSummariesWithReport		Y	
UseIndexForSpeed	Y		
TranslateDOSStrings	Y		
TranslateDOSMemos	Y		
ConvertDateTimeType	Y		
UseCaseInsensitiveSQLData	Y		
VerifyOnEveryPrint	Y		
UseDummyData		Y	
ConvertNullFieldToDefault	Y		
MorePrintEngineErrorMessages	Y		

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
DontGenerateDataForHiddenObjects	Y		
Summary Info			
			Unchanged
Formula Field			
Create	Y		
Properties:			
Syntax	RW		
Group Name Field			
Properties:			
Group		R	
GroupNumber	R		
GroupNameConditionFormula	RW		
Parameter Field			
Create	Y		
Properties:			
PlaceInGroup	RW		
GroupNumber	RW		
EnableExclusiveGroup	RW		
RunningTotal Field			
Create	Y		
Properties:			
RunningTotalName	R		
Group		R	
Operation	RW	R	
OperationParameter	RW	R	
SecondarySummarizedField	RW	R	
SummarizedField	RW	R	
EvaluateCondition	RW		
evalGroupNumber	RW		
ResetCondition	RW		
resetGroupNumber	RW		
EvaluateConditionField	RW		
ResetConditionField	RW		
EvaluateConditionFormula	RW		
ResetConditionFormula	RW		
HierarchicalSummaryType	RW		
Special Field			
Create	Y		

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
Properties:			
SpecialVarType	R	RW	
SQLExpression Field			
Create	Y		
Properties:			
SQLExpressionFieldName	R		
Text	RW	R	
Summary Field			
Create	Y		
Properties:			
SummaryType	RW	R	
SummarizedField	RW	R	
SecondarySummarizedField	RW	R	
SummaryOperationParameter	RW	R	
HierarchicalSummaryType	RW		
Group		R	
Custom function			
			Unchanged
Groups			
Add	Y		
Properties:			
ConditionField	RW	R	
GroupOptions	RW	R	
repeatGroupHeader	Y		
keepGroupTogether	Y		
TopOrBottomNGroups	Y		
topOrBottomNSortFieldName	Y		
hierarchicalSorting	Y		
instanceIDField	Y		
parentIDField	Y		
groupIndent	Y		
customizeGroupName	Y		
Metadata			
View Metadata reports (Set Location)	Y		
Repository			
			Unchanged
OLAP			
			Unchanged

Feature comparison: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comments**
Templates			
			Unchanged
Report Alerts			
Create	Y		
Properties:			
Name	RW		
Enable	RW		
Message	RW		
MessageFormula	RW		
ConditionFormula	RW		
nTriggeredInstances	RW		
alertConditionFormulaSyntax	RW		
alertMessageFormulaSyntax	RW		
FromMainDocument	RW		
alertReportName	RW		
Report Parts			
			Unchanged
Creating Report			
			Unchanged
Drill Down			
Map	Y		
Report part		Y	
CE Integration			
Log on CMS		Y	
Open report from CMS, save report to CMS		Y	
Search Expert			
Find Strings, Number, Date	Y		
Search backward	Y		
Events			
NoData	Y		
BeforeFormatPage	Y		
AfterFormatPage	Y		
FieldMapping	Y		

Functional matrix: RDC and .NET assemblies			
Features	**RDC**	**.NET**	**Comment**
Productivity			
			Unchanged
Viewing			
Report Page Viewer		X	
Report Parts Viewer		X	
Winform Viewer		X	
Webform Viewer		X	
Mobile Viewer		X	
Report Modification			
			Unchanged
Report Creation			
Report Creation API	X		
Embeddable Report Designer	X		
Report Processing			
Report part support		X	
.NET UFL support		X	

Finding more information

Accompanying RDC and .NET applications

http://support.businessobjects.com/communityCS/FilesAndUpdates/cr_xi2_migrating_from_rdc_to_net_samples.zip.asp

Crystal Reports XI .NET Deployment

http://support.businessobjects.com/communityCS/TechnicalPapers/crxi_net_deployment.pdf.asp

Merge Modules

http://support.businessobjects.com/fix/merge_modules.asp

Sample applications

http://support.businessobjects.com/fix/samplescr.asp#04

Technical Support

http://www.businessobjects.com/contact/support.asp

www.businessobjects.com