Chapter 12 Previewing the Report at Runtime

Previewing a report is probably what users do most and Crystal Reports provides excellent preview capabilities for your application.

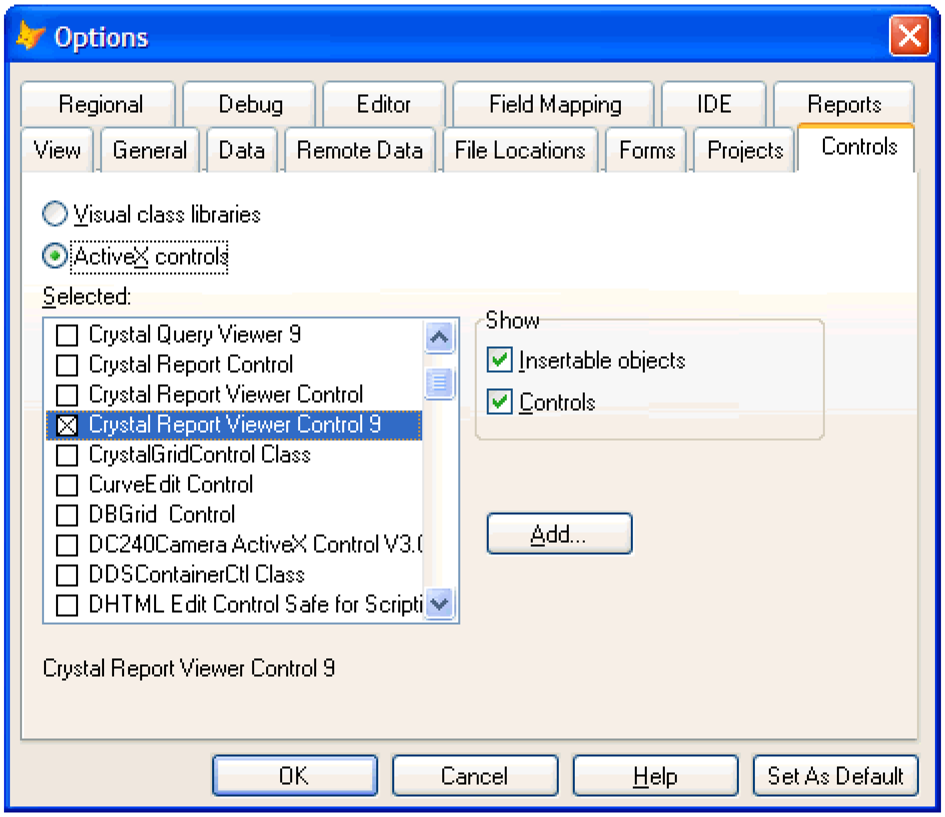
As you have seen in previous chapters, the Crystal Reports Report Design Component (RDC) gives you total control over creating, modifying, printing, and exporting reports in your application. However, you have not seen how to preview the report.

Beginning with Crystal Reports 9, Crystal Decisions began shipping two versions of the report viewer. The first is an ActiveX control for COM-based development environments. The second is a Java Bean control for use in Java. This chapter addresses using the ActiveX viewer. You will soon see how to control most aspects of using this control and how it displays. The Viewer Control is also multi-threaded. This means the user can start previewing the report before all the data loads.

Registering the control

To use an ActiveX control on your Visual FoxPro form, you need to list it on the ActiveX Form Control toolbar. The following steps walk you through this process.

1. Select Tools | Options from the menu. The Options dialog box displays (see **Figure 1**).
2. Select the Controls tab.
3. Select ActiveX controls from the option group.
4. Select Crystal Report Viewer Control 9 from the list. Make sure the box has an X in it.
5. Click Set As Default so Visual FoxPro remembers the setting.
6. Click OK to close the Options dialog box.



( ) Visual class libraries
(●) ActiveX controls

Selected:

- [ ] Crystal Query Viewer 9
- [ ] Crystal Report Control
- [ ] Crystal Report Viewer Control
- [x] Crystal Report Viewer Control 9
- [ ] CrystalGridControl Class
- [ ] CurveEdit Control
- [ ] DBGrid Control
- [ ] DC240Camera ActiveX Control V3.(
- [ ] DDSContainerCtl Class
- [ ] DHTML Edit Control Safe for Scripti

Show

- [✓] Insertable objects
- [✓] Controls

Crystal Report Viewer Control 9

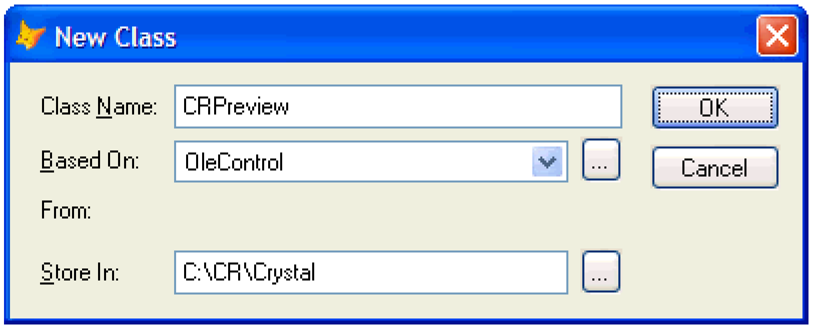
***Figure 1**. Select Crystal Report Viewer Control in the Options dialog box to make it available.*

Creating a preview form

With the Crystal Report Viewer Control available to your application, you can create a preview form. One of the unique features of Visual FoxPro is the ability to subclass an ActiveX control. This allows you to add functionality to the control and reuse it in other places. For example, in Chapter 13, “The Report Designer Control,” you see how to use the Embeddable Report Design Control and use the preview control to build full report design capabilities.

The following steps walk you through creating the form.

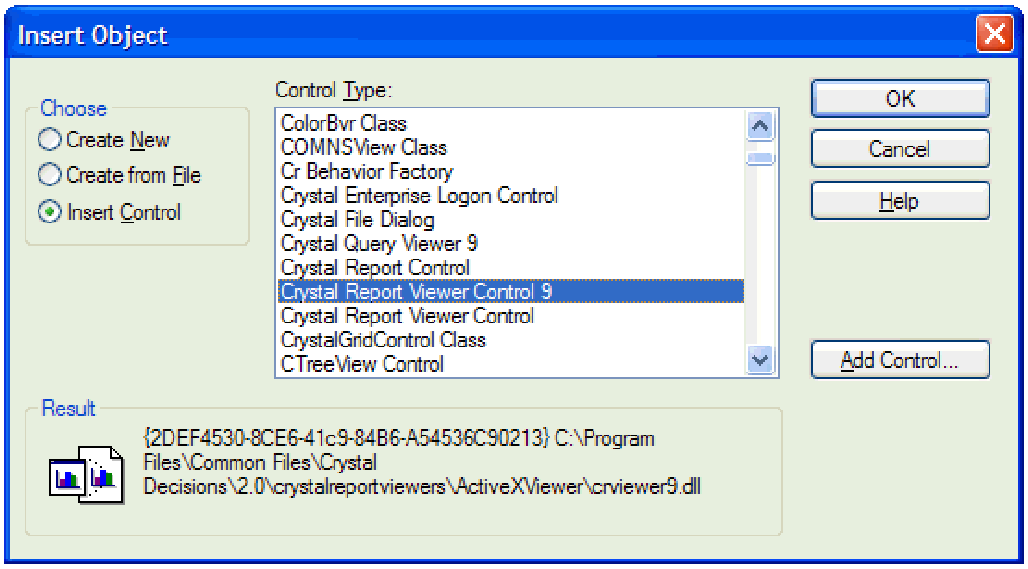
1. Type “CREATE CLASS” in the Command Window. The New Class dialog box displays (see **Figure 2**).



Class Name: CRPreview
Based On: OleControl
From:
Store In: C:\CR\Crystal

Figure 2. Use the New Class dialog box to define new classes.

2. Enter CRPreview for the Class Name.
3. Select OleControl for Based On.
4. Enter Crystal for the name of the Visual Class Library in the Store In field.
5. Click OK. The Insert Object dialog box displays (see **Figure 3**).



Choose
Create New
Create from File
[✓ Insert Control]

Control Type:
ColorBvr Class
COMNSView Class
Cr Behavior Factory
Crystal Enterprise Logon Control
Crystal File Dialog
Crystal Query Viewer 9
Crystal Report Control
[✓ Crystal Report Viewer Control 9]
Crystal Report Viewer Control
CrystalGridControl Class
CTreeView Control

Result
{2DEF4530-8CE6-41c9-84B6-A54536C90213} C:\Program Files\Common Files\Crystal Decisions\2.0\crystalreportviewers\ActiveXViewer\crviewer9.dll

Figure 3. Use the Insert Object dialog box to select the ActiveX control to subclass.

6. Select Insert Control in the Choose option.
7. Select Crystal Report Viewer Control 9 in the Control Type list, and then click OK. The Class Designer displays.

8. Add two properties to the class. The first, oCrystal holds a reference to the Application object. The second, oReport, references the report object. The Report Viewer control uses the RDC automation server for many of its own services such as printing and exporting.
9. Add a new method called Setup. Call this method to do all the set up needed by the viewer control when using it on a form.

If you look at the property sheet, you see many properties, events, and methods for the Viewer Control. It may be tempting to change the properties using the property sheet. However, this doesn't work in Visual FoxPro. You need to set the properties in code. **Table 1** lists the properties.

***Table 1**. This table contains the properties of the Viewer Control.*

Property	Type	Read Only	Description
ActiveViewIndex	Numeric	Y	Gets the index of the current view tab.
DisplayBackgroundEdge	Logical	N	If True, displays the background edge. This offsets the report from the edge of the control.
DisplayBorder	Logical	N	If True, displays the border of the viewer.
DisplayGroupTree	Logical	N	If True, displays the group tree.
DisplayTabs	Logical	N	If True, displays the tabs for different views.
DisplayToolbar	Logical	N	If True, displays the toolbar.
EnableAnimationCtrl	Logical	N	If True, displays the animation control.
EnableCloseButton	Logical	N	If True, displays the Close button.
EnableDrillDown	Logical	N	If True, enables drill down.
EnableExportButton	Logical	N	If True, displays the export button.
EnableGroupTree	Logical	N	If True, displays the group tree.
EnableHelpButton	Logical	N	If True, displays the help button.
EnableNavigationControls	Logical	N	If True, displays the page navigation controls.
EnablePopupMenu	Logical	N	If True, enables the popup menu.
EnablePrintButton	Logical	N	If True, makes the print button visible.
EnableProgressControl	Logical	N	If True, displays the progress control.
EnableRefreshButton	Logical	N	If True, makes the refresh button visible.
EnableSearchControl	Logical	N	If True, makes the Search button visible and enables the search feature.
EnableSearchExpertButton	Logical	N	If True, makes the search expert button visible.
EnableStopButton	Logical	N	If True, displays the stop button.
EnableToolbar	Logical	N	If True, the toolbar is functional.
EnableZoomControl	Logical	N	If True, makes the zoom drop down control visible.
IsBusy	Logical	Y	Returns the busy status of the control.
ReportSource	Object	N	A reference to a report object.
TrackCursorInfo	Object	Y	A reference to the track cursor info object, discussed later in this chapter.
ViewCount	Numeric	Y	Returns the number of view tabs currently available.

The only property that needs additional explanation is the TrackCursorInfo property. This property holds a reference to the TrackCursorInfo object containing information about the mouse cursor used for the Viewer control. Using this object, you can get or set the mouse cursor type. **Table 2** lists the properties of the TrackCursorInfo object.

Table 2*. This table lists the properties of the TrackCursorInfo object.*

Property	**Type**	**Read Only**	**Description**

DetailAreaCursor	Numeric	N	Mouse cursor to use for the detail area of the viewer control. Possible values:
			crAppStartingCursor = 12
			crArrowCursor = 1
			crCrossCursor = 2
			crDefaultCursor = 0
			crHelpCursor = 13
			crIBeamCursor = 3
			crMagnifyCursor = 99
			crNoCursor = 10
			crWaitCursor = 11

DetailAreaFieldCursor	Numeric	N	Mouse cursor to use for the detail fields of the viewer control. Values are the same as for the DetailAreaCursor.
GraphCursor	Numeric	N	Mouse cursor to use for a graph displayed in the viewer control. Values are the same as for the DetailAreaCursor.
GroupAreaCursor	Numeric	N	Mouse cursor to use for a group header or footer in the viewer control. Values are the same as for the DetailAreaCursor.
GroupAreaFieldCursor	Numeric	N	Mouse cursor to use for a field in a group header or footer in the viewer control. Values are the same as for the DetailAreaCursor.

The TrackCursorInfo object does not have any methods or events.

The ReportSource property is the most important property of the Viewer control. You need to set it to a Report Object before you can use the viewer. Do this in the Setup method of the subclassed control. Because this control is dropped on a form or another container, and the control instantiates before the parent container, you can’t use the Init method to accept the necessary parameters. So, add the following code to the Setup method:

```
LPARAMETERS tcReport, toReport, toCrystal

WITH This
  IF PARAMETERS() = 1
    * Only the report name was passed in. Create the CR object
    .oCrystal = CREATEOBJECT("CrystalRuntime.Application")
    .oReport = .oCrystal.OpenReport(tcReport)
  ELSE
    * All the parameters were passed in. Use those references to the CR objects
    .oCrystal = toCrystal
    .oReport = toReport
  ENDIF
  * Setup the viewer properties
  .ReportSource = .oReport
  .DisplayBorder = .F.
  .DisplayBackgroundEdge = .T.
  .EnableProgressControl = .F.
  .Resize()
  .ViewReport()
ENDWITH
```

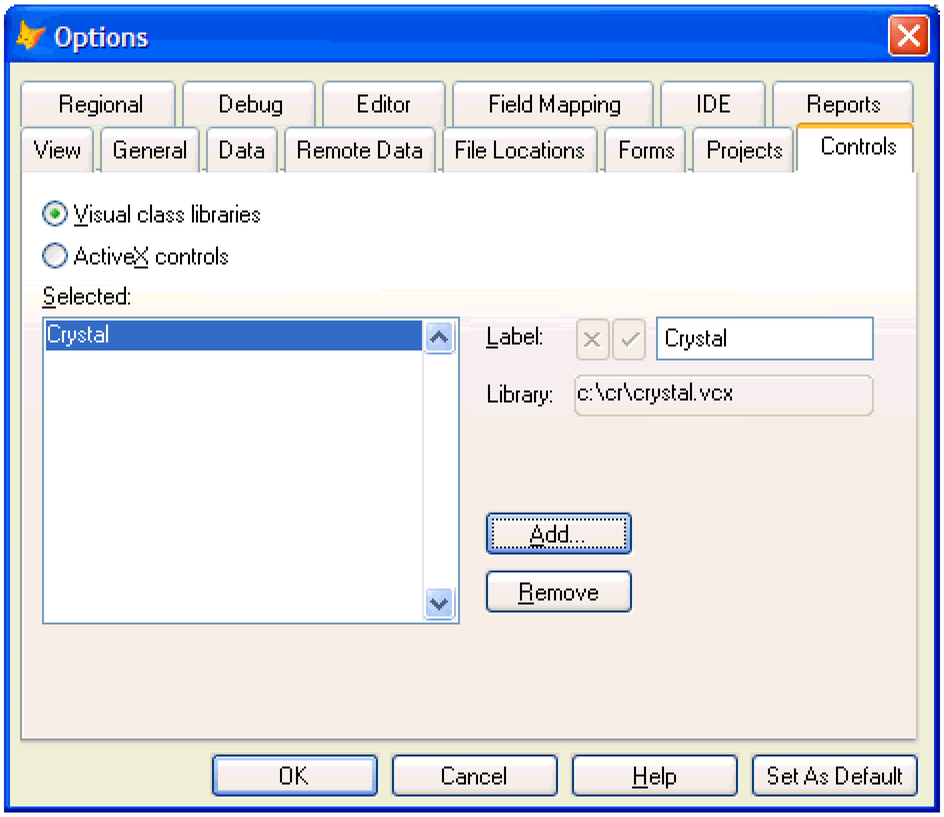
The ViewReport method actually loads the data and displays the report. I explain the methods of the Viewer control later in this chapter. It is important to call the Resize method so the control is the same size as the container you drop it on. Here's the code for the Resize method:

```
WITH This
  .Top = 0
  .Left = 0

  IF .Parent.BaseClass = "Page"
    * The viewer is on the page of a page frame.
    * Set the height and width the same as the page
    .Height = .Parent.Parent.PageHeight
    .Width = .Parent.Parent.PageWidth
  ELSE
    * Just use the height and width of the parent
    .Height = .Parent.Height
    .Width = .Parent.Width
  ENDIF
ENDWITH
```

The control is now ready to use. Close the class designer and save changes to the control. Now, with a customized version of the Viewer Control, you need to make it available to the Visual FoxPro form designer.

1. Select Tools | Options from the VFP menu. The Options dialog box displays.
2. Select the Controls page (see **Figure 4**).



[✓ Visual class libraries] ActiveX controls

Selected: Crystal

Label: Crystal

Library: c:\cr\crystal.vcx

***Figure 4**. Make the subclassed Viewer control available to the form designer by selecting it on the Controls page of the Options dialog box.*

3. Select Visual class libraries from the option group.
4. Click Add, select Crystal from the Open dialog box, and then click Open.
5. Back in the Options dialog box, click Set as Default and then OK. This saves your changes.

Finally, it's time to create the preview form.

1. Create a new form and call it CrystalPreview. The Form Designer displays.
2. Click View Classes on the Form Controls toolbar and select Crystal from the shortcut menu. The Forms Controls toolbar changes to reflect the controls available in the Crystal class library (see **Figure 5**).

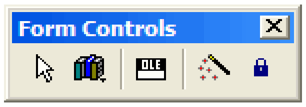


***Figure 5**. Configure the Form Controls toolbar to show the Crystal class library.*

3. Click the CRPreview OLE control button on the Form Controls toolbar and drop it onto the form. The CRPreview control appears as a square. Don't worry about sizing the control. Form code handles this.
4. On the Properties sheet, change the name of the control from CRPreview1 to oleCRPreview.
5. Double-click the form (not the control) and enter the following code into the Init method of the form:

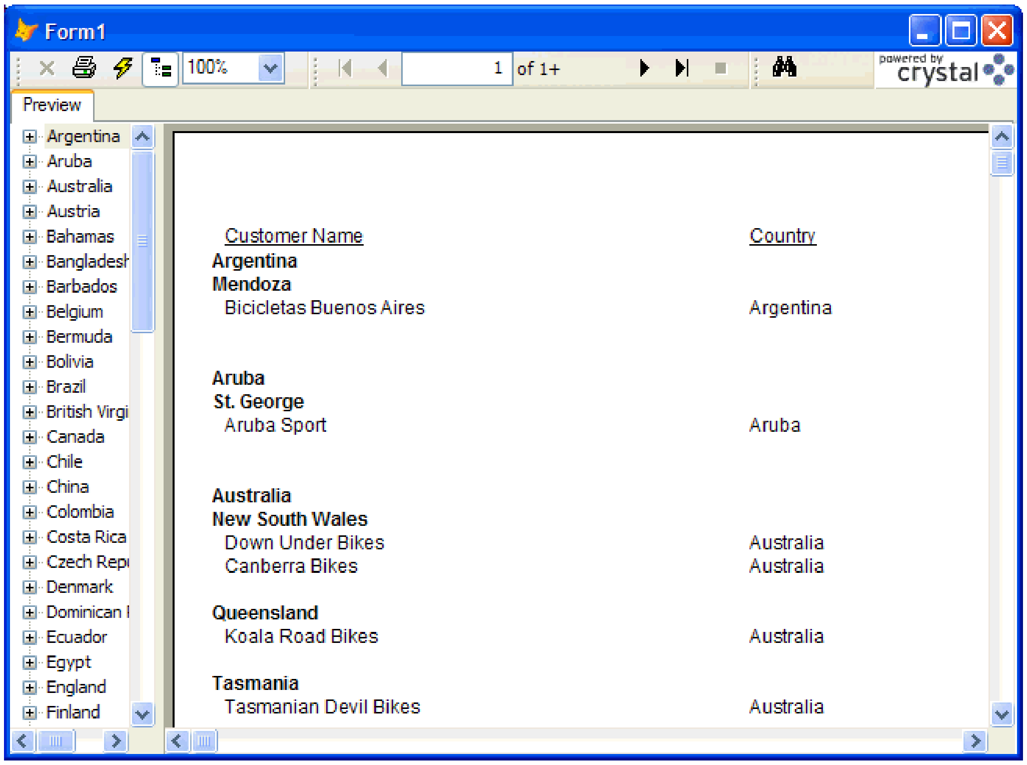
```
LPARAMETERS tcReportName
This.oleCRPreview.Setup(tcReportName)
```

6. Enter the following code in the Resize method of the form. This causes the viewer control to resize when you resize the form.

```
This.oleCRPreview.Resize()
```

7. Close the form and save the changes.
8. Time to test the form. You do this from the Command Window. Substitute the report name for one you have available. The preview is shown in **Figure 6**.

```
DO FORM CrystalPreview WITH "C:\CR\MyReport.RPT"
```



```
Customer Name                                  Country
Argentina
Mendoza
 Bicicletas Buenos Aires                       Argentina

Aruba
St. George
 Aruba Sport                                   Aruba

Australia
New South Wales
 Down Under Bikes                              Australia
 Canberra Bikes                                Australia

Queensland
 Koala Road Bikes                              Australia

Tasmania
 Tasmanian Devil Bikes                         Australia
```

***Figure 6**. Preview of the finished report.*

If you are using a data source that requires you to log on, you need to release the Report object from the Report Viewer before you can log off. This is because the Report object, not the Viewer Control, holds the data. There are two ways to release the Report object. First, you assign a new report object to the viewer or you destroy the viewer object.

There you have it! A Crystal Report viewer for your application. However, there are many methods and events available on the control. The rest of this chapter discusses them.

Methods of the Viewer Control

The Crystal Reports Viewer Control has several methods you can call to affect the preview. You call many of the methods from the control's native toolbar. This gives you the ability to create your own toolbar for the viewer and still give full functionality to the end user.

Showing the report

The first method you should call is Refresh. This ensures you have current data loaded into the report before it is shown. The Refresh method has no parameters.

```
ThisForm.oleViewer.Refresh()
```

The ViewReport method shows the report. This method has no parameters.

```
ThisForm.oleViewer.ViewReport()
```

Once the report is visible, you can change the way the report displays. You can do things such as navigate to a specific page, change the zoom, or drill down to a specific group either on its own tab (called a view) or in the main preview window. It's also possible to search for specific text.

You can also navigate to a relative page in the report. The following method calls show how to do this. None of the methods take parameters.

```
ThisForm.oleViewer.ShowFirstPage()
ThisForm.oleViewer.ShowLastPage()
ThisForm.oleViewer.ShowPreviousPage()
ThisForm.oleViewer.ShowNextPage()
```

You display a specific page with the ShowNthPage method. Here is the syntax:

```
ShowNthPage(nPage)
```

nPage	Numeric	The page number of the page to show.

The Zoom method zooms in or out of the displayed report.

```
Zoom(nValue)
```

nValue	Numeric	The zoom percentage. For example, 400 zooms the report to 400%. You can also pass 1 to fill the width of the view window or 2 to fit the entire page in the window.

Print a report with the PrintReport method, which has no parameters. Here is an example:

```
ThisForm.oleViewer.PrintReport()
```

Working with groups

Groups are supported in a couple of different ways. You can show a specific group in the current view or add a new view for a group as a drill down.

The ShowGroup method displays a group in the current view.

```
ShowGroup(cGroupPath | aGroupPath)
```

cGroupPath	Character	A colon separated string containing the path to the group. For example, USA:Utah:Salt Lake City
aGroupPath	Array	An array containing the path for the group.

Instead of navigating to a particular group in the current view, you can add a new view with the AddView method.

```
AddView(cGroupPath | aGroupPath)
```

cGroupPath	Character	A colon separated string containing the path to the group for the new view. For example, USA:Utah:Salt Lake City
aGroupPath	Array	An array containing the path for the new view.

You activate a particular view with the ActivateView method. The parameter is the view number you want to activate. View one is always the primary view.

```
ActivateView(nView)
```

nView	Numeric	The view number to activate. Each view is a separate tab on the control.

Use CloseView to close a specific view. View one is the Preview itself. If you try to close it, you get an error.

```
CloseView(nView)
```

nVew	Numeric	The view number to close.

You retrieve the name of the view with the GetViewName method.

```
GetViewName(cTab)
```

cTab	Character	The name of the tab containing the view.

Finally, you get the path to the view with the GetViewPath method. GetViewPath method returns an array.

```
GetViewPath([nView])
```

nView	Numeric	The view number containing the path.

Retrieving information

The Crystal Reports Viewer has three methods you can use to retrieve specific information. The first is GetCurrentPageNumber. This method returns the page number displayed in the current view. It does not have any parameters.

```
GetCurrentPageNumber()
```

You locate specific information in a report with the SearchForText method. If the specified text is found, it is highlighted in the current view. If the text is not found, the message Search could not find any more instances of the specified text after this page displays. The search always begins on the current page, searches forward in a report, and is always specific to the current view. Here’s the syntax:

```
SearchForText(cText)
```

cText	Character	The text to search for.

Finally, SearchByFormula allows you to enter a search formula. For example, you can search for Country = 'USA' by using the following formula:

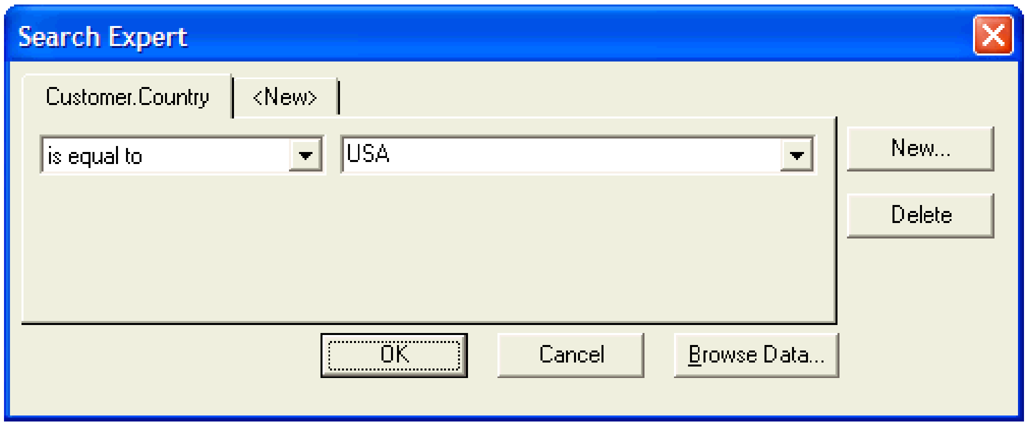
```
{Customer.Country} = 'USA'
```

Here's the syntax:

```
SearchByFormula(cFormula)
```

cFormula	Character	The formula you want to use for the search.

When you pass an empty string to the SeachbyFormula method, the Viewer control displays the Search Expert, where you enter your search criteria (see **Figure 7**).



is equal to USA

Figure 7*. The Search Expert displays when you pass an empty string to the SearchbyFormula method.*

Events

Events occur when something happens with the application. For example, clicking a particular field causes the click event to fire. Unlike methods, you don't pass parameters to events. Instead, the Crystal Report Viewer Control passes parameters to code you write in event methods.

In the event method code, you can run any code you want. You can even cause the default behavior of the event to not occur. All the events have the lDefault parameter. If you set this parameter to False, the default behavior does not occur.

Report objects events

Report object events occur when you click or double-click a field, heading, or label in a report. Doing this creates the EventInfo object. This object contains information about the event and passes as a parameter to the event method. **Table 3** lists the EventInfo object properties.

***Table 3**. This table shows the properties of the EventInfo object.*

Property	Type	Read Only	Description
CanDrillDown	Logical	Y	If True, you can drill down on the object.
Index	Numeric	Y	The index of the control in a control array.
ParentIndex	Numeric	Y	Gets a reference to the object's parent's index.
Text	Character	Y	The object's text string.

Type	Numeric	Y	The type of object. Possible values:
			crBitMap = 103
			crBlob = 104
			crBox = 106
			crCrosstab = 110
			crCrosstabChart = 108
			crCrosstabMap = 115
			crDatabaseFieldType = 1
			crDetailChart = 109
			crDetailMap = 116
			crDetailSection = 202
			crFormulaFieldType = 5
			crGraphic = 111
			crGroupChart = 107
			crGroupFooterSection = 201
			crGroupHeaderSection = 200
			crGroupMap = 114
			crGroupNameFieldType = 8
			crLine = 105
			crOLAPChart = 113
			crOLAPCrosstabFieldType = 4
			crOLAPDataFieldType = 3
			crOLAPDimensionFieldType = 2
			crOLAPMap = 117
			crOLEObject = 101
			crOOPSubreport = 112
			crPageFooterSection = 206
			crPageHeaderSection = 203
			crSpecialVarFieldType = 7
			crSubreport = 102
			crSummaryFieldType = 6
			crText = 100
			crUnknownFieldDefType = 0

The EventInfo object has a single method, GetFields, which returns a Fields Collection. GetFields does not have any parameters.

```
oFields = oEventInfo.GetFields()
```

The Fields collection holds a collection of field objects and has no methods. **Table 4** lists the properties of this collection.

***Table 4**. This table lists the properties of the Fields collection.*

Property	Type	Read Only	Description
Count	Numeric	Y	The total number of items in the collection.
Item	Numeric	Y	Index of the item in the collection.
SelectedFieldIndex	Numeric	Y	Index of the item being selected. This is the item number of the clicked field in the viewer.

If the user clicks a field in the viewer, the following code shows how to reference the Field object to get information about the field.

```
oFields = oEventInfo.GetFields()
oField = oFields.Item(oFields.SelectedFieldIndex)
```

The Field object has no methods or events. **Table 5** lists its properties.

***Table 5**. This table contains the properties of the Field object.*

Property	Type	Read Only	Description
FieldType	Numeric	Y	The field type. Possible values: crBoolean = 5; crCurrency = 4; crDate = 6; crDateTime = 8; crInt16 = 1; crInt32 = 2; crInt8 = 0; crNumber = 3; crString = 9; crTime = 7; crUnknownFieldType = 255
IsRawData	Logical	Y	.T. if the data is raw data.
Name	Character	Y	The name of the field.
Value	Variant	Y	The value of the field.

Getting back to the report object, there are two events that fire. The first is Clicked, which occurs when you click an object.

```
Clicked(nX, nY, oEventInfo, lDefault)
```

nX	Numeric	The x coordinate of the object clicked.
nY	Numeric	The y coordinate of the object clicked.
oEventInfo	Object	An object containing information about the clicked report object. See Table 3 for a list of properties for this object.
lDefault	Logical	If True, performs the default action of the event.

The second is DblClicked, which fires when you double-click an object.

`DblClicked(nX, nY, oEventInfo, lDefault)`

nX	Numeric	The x coordinate of the object that was double-clicked.
nY	Numeric	The y coordinate of the object that was double-clicked.
oEventInfo	Object	An object containing information about the clicked report object. See Table 3 for a list of properties for this object.
lDefault	Logical	If True, performs the default action of the event.

Drill events

Drill events fire when you double-click an object to drill down.

`DrillOnDetail(aValues, nIndex, lDefault)`

aValues	Array	An array of objects containing details on the fields.
nIndex	Numeric	The index of the array for the field that was actually clicked
lDefault	Logical	If True, performs the drill down.

`DrillOnGraph(nPage, nX, nY, lDefault)`

nPage	Numeric	The page number of the report containing the graph.
nX	Numeric	The X coordinate of the graph that received the click.
nY	Numeric	The Y coordinate of the graph that received the click.
lDefault	Boolen	If True, performs the drill down.

`DrillOnGroup(aGroup, nDrillType, lDefault)`

aGroup	Array	An array containing all the group names for the drilled-down group.	
nDrillType	Numeric	The type of drill down. Possible values:	

		LoadingNothing = 0
		LoadingPages = 1
			LoadingQueryInfo = 3
			LoadingTotaller = 4

lDefault	Boolen	If True, performs the drill down.	

`DrillOnSubreport(aGroup, cSubreport, cTitle, nPage, nIndex, lDefault)`

aGroup	Array	An array containing all the group names for the drilled-down group.
cSubreport	Character	The name of the subreport.
cTitle	Character	The title of the subreport.
nPage	Numeric	The page containing the subreport.
nIndex	Numeric	The index of the drilled-down subreport.
lDefault	Logical	If True, performs the drill down.

Toolbar objects events

Toolbar object events trigger when you click a control on the toolbar. Here are the event methods available. In most cases, the only parameter is lDefault.

`FirstPageButtonClicked(lDefault)`

lDefault	Logical	If True, displays the first page.

`LastPageButtonClicked(lDefault)`

lDefault	Logical	If True, displays the last page.

`NextPageButtonClicked(lDefault)`

lDefault	Logical	If True, displays the next page.

`PrevPageButtonClicked(lDefault)`

lDefault	Logical	If True, displays the previous page.

`GotoPageNClicked(lDefault, nPage)`

lDefault	Logical	If True, displays the Nth page.
nPage	Numeric	The page number to display

`CloseButtonClicked(lDefault)`

lDefault	Logical	If True, closes the current view.

`ExportButtonClicked(lDefault)`

lDefault	Logical	If True, performs the export.

`GroupTreeButtonClicked(lVisible)`

lVisible	Logical	If True, makes the group tree visible.

`PrintButtonClicked(lDefault)`

lDefault	Logical	If True, prints the report.

`RefreshButtonClicked(lDefault)`

lDefault	Logical	If True, refreshes the report data.

`SearchButtonClicked(cExpression, lDefault)`

cExpression	Character	The expression to search for.
lDefault	Logical	If True, performs the search.

`SearchExpertButtonClicked(lDefault)`

lDefault	Logical	If True, displays the search expert.

`StopButtonClicked(nLoadType, lDefault)`

nLoadType	Numeric	The type of data being loaded into the report.
lDefault	Logical	If True, stops the data load process.

`HelpButtonClicked()`

Note the HelpButtonClicked event does not send any parameters.

Miscellaneous events

Finally, there are several other events that don't fit neatly into other categories. The first is the DownloadStarted event. This event fires when Crystal Reports starts downloading data into a report.

`DownloadStarted(nLoadType)`

nLoadType	Numeric	The type of data being loaded into the report. Possible values:
		LoadingNothing = 0 LoadingQueryInfo = 3
		LoadingPages = 1 LoadingTotaller = 4

Once Crystal Reports completes the download, it triggers the DownloadFinished event.

`DownloadFinished(nLoadType)`

nLoadType	Numeric	The type of data being loaded into the report. Possible values:
		LoadingNothing = 0 LoadingQueryInfo = 3
		LoadingPages = 1 LoadingTotaller = 4

When the user clicks the Selection Formula button, it fires the SelectionForumulaButtonClicked event. This event is only valid when the viewer is assigned a report source from the RDC. In other words, you will probably never use it.

`SelectionFormulaButtonClicked(cFormula, lUseDefault)`

cFormula	Character	The current selection formula. This is replaced if the user selects anything.
lUseDefault	Logical	If True, the default action occurs.

The SelectionFormulaBuilt event fires when a new selection formula is applied to a report.

`SelectionFormulaBuilt(cFormula, lDefault)`

cFormula	Character	The selection formula applied to the report.
lDefault	Logical	If True, performs the default action.

Clicking a particular group in the group tree triggers the ShowGroup event.

`ShowGroup(aGroup, lDefault)`

aGroup	Array	An array containing the group list.
lDefault	Logical	If True, displays the group.

Changing from one view to another causes the ViewChanging event to fire, notifying you that the view is about to change.

`ViewChanging(nOldView, nNewiew)`

nOldView	Numeric	The number of the old view.
nNewView	Numeric	The number of the new view.

Once the view changes, the ViewChanged event occurs.

`ViewChanged(nOldView, nNewView)`

nOldView	Numeric	The number of the old view.
nNewView	Numeric	The number of the new view.

Changing the Zoom level triggers the ZoomLevelChanged event.

`ZoomLevelChanged(nLevel)`

nLevel	Numeric	The new zoom level.

Finally, if an error occurs when you set the Viewer's ReportSource property or for some reason the report source cannot load, the OnReportSourceError event fires.

`OnReportSourceError(cMessage, nCode, lDefault)`

cMessage	Character	The text of the error message.
nCode	Numeric	The error number.
lDefault	Logical	If True, performs the default action.

Summary

This chapter shows you how to create a report preview form and how to control the preview itself. Crystal Reports provides many capabilities for controlling the preview process. You will use the preview control again in Chapter 13, "The Report Designer Control."

Updates and corrections to this chapter can be found on Hentzenwerke's web site, **www.hentzenwerke.com**. Click "Catalog" and navigate to the page for this book