The Crystal Custom Control

Overview

- Visual Basic is an extensible programming language: it has been designed so that developers can readily add new capabilities as they see the need. One of the ways the program can be extended is through custom controls. Custom controls are programs that add tools to the Visual Basic toolbox. Once in the toolbox, these tools can be used on any form just as if they were part of the original Visual Basic language.

- There are two types of Custom Controls:

1) Custom controls that can produce objects that are visible at design time and at runtime.

2) Custom controls that can also produce objects that are visible at design time but invisible at runtime. These kinds of objects were designed to make it easier for your applications to access the capabilities available in Dynamic Link Libraries (DLL's). By setting the properties for these objects in the Properties dialog box, you eliminate the need for a lot of manual coding.

Crystal Reports Professional Edition comes with a custom control in the second category:

- You receive a free runtime license for this DLL so you can include it without charge with each copy of your application.

- The Crystal Custom Control is a set of tools that makes it easy for you to build the connection between your application and the print engine.

- You can't use the Crystal Custom Control and make direct calls to print engine functions at the same time.

Adding the Crystal Custom Control to your project

You add the Crystal Custom Control to your project using the File|Add File command on the Visual Basic menu bar.

NOTE: CRYSTAL.VBX can be added to AUTOLOAD.MAK to automatically load the Custom Control to your project.

Getting Help

You can find information related to the Crystal Custom Control in the Help File under Developer's Reference, in the Developer's Reference Manual starting on P 5-1 and in a sample VB application that ships with CRW 4.0 in a subdirectory called VBXDemo(VBXDemo.MAK).

When actually using the Crystal Custom Control you can highlight a property and select F1 to automatically open the help file which will take you directly to the definition of the respective VBX property.

Error Messages

A list of Error messages for the Crystal Custom Control can be found in the CRW 4.0 Help file by searching on ERROR MESSAGES, CRYSTAL CUSTOM CONTROL.

Using the Crystal Custom Control

Once you have the Crystal Custom Control object on your form, you build the connection between your application and Crystal Reports by setting the object's properties via the control's Properties list.

Design Time Properties:

About	AutoDesign	BoundReportFooter
BoundReportHeading	Connect	CopiesToPrinter
DataSource	Destination	DetailCopies
DiscardSavedData	EMailCCList	EMailMessage
EMailSubject	EMailToList	EMailVIMBCCList
GroupSelectionFormula	Index	Left
MarginBottom	MarginLeft	MarginRight
MarginTop	Name	PrinterCollation
PrintCopies	PrinterDriver	PrinterName
PrinterPort	PrinterStartPage	PrinterStopPage
PrintFileCharSepQuote	PrintFileCharSepSeparator	PrintFileName
PrintFileType	PrintFileUseRptDateFmt	PrintFileUseRptNumberFmt
ReportFileName	ReportSource	ReportTitle
SelectionFormula	SQLQuery	Tag
Top	WindowBorderStyle	WindowControlBox
WindowControls	WindowHeight	WindowLeft
WindowMaxButton	WindowMinButton	WindowState
WindowTitle	WindowTop	WindowWidth

Runtime Properties:

Action	Connect	CopiesToPrinter
DataFiles	Destination	DetailCopies
DiscardSavedData	EMailCCList	EMailMessage
EMailSubject	EMailToList	EMailVIMBCCList
Formulas	GraphData	GraphOptions
GraphText	GraphType	GroupCondition
GroupSelectionFormula	GroupSortFields	LastErrorNumber
LastErrorString	Left	MarginBottom
MarginLeft	MarginRight	MarginTop
Name	Password	PrintDay

PrinterCollation	PrinterCopies	PrinterDriver
PrinterName	PrinterPort	PrinterStartPage
PrinterStopPage	PrintFileCharSepQuote	PrintFileCharSepSeparator
PrintFileName	PrintFileType	PrintFileUseRptDateFmt
	PrintMonth	PrintReport
PrintYear	RecordsPrinted	RecordsRead
RecordsSelected	ReportDisplayPage	ReportFileName
ReportLatestPage	ReportStartPage	ReportTitle
SectionFont	SectionFormat	SectionLineHeight
SectionMinHeight	SelectionFormula	SessionHandle
SortFields	SQLQuery	Status
StoredProcParam	Top	UserName
WindowBorderStyle	WindowControlBox	WindowControls
WindowHeight	WindowLeft	WindowMaxButton
WindowMinButton	WindowParentHandle	WindowState
WindowTitle	WindowTop	WindowWidth

ReportFileName

```
If Main.Report1.PrintReport <> 0 Then
     Unload C
     Screen.MousePointer = 0
     MsgBox "Printing the Report to Window has caused Error#: " &
Main.Report1.LastErrorNumber & " - " & Main.Report1.LastErrorString
Exit Sub
Else
    Main!StatusBar.Caption = "Preview to window Successful."
End if
```

Destination

```
Main.Report1.Destination = 0

‘ 0 Print to Window
‘ 1 Print to Printer
‘ 2 Print to File
```

SelectionFormulas

Setting Record Selections at Runtime

If are using a *string* in your record selection criteria, it might look like:

```
Sub Command1_Click ()
  'if we hard-code the name
  Report1.SelectionFormula = "{file.field} = 'Bob Brown'"
  'If we use the value of the Textbox
  Text1.Text = "Bob Brown"
```

```
    Recselect$ = "{file.field} ='" & Text1.Text & "'"
    MsgBox Recselect$
    'Should display  {file.field} = 'Bob Brown'
    'The result of a message box should look exactly like the record selection if it
   'were built in Crystal Reports itself.
    Report1.SelectionFormula = "{file.field} ='" & Text1.Text & "'"
    Report1.action = 1
End Sub
```

If you are using a string with an apostrophe within it, it might look like:

```
 Sub Command1_Click ()
   'if we hard-code the name
   RecSelect$ = "{file.field} = ""My mother's name"""
   MsgBox RecSelect$
  'If we use the value of the Textbox
   Text1.Text = "Bob Brown's House"
    Recselect$ ="{table1.text} = """ & Text1.Text & """"
    MsgBox Recselect$
    'Should display  {file.field} = "Bob Brown's House"'
    'The result of a message box should look exactly like the record selection if it
   'were built in Crystal Reports itself.
    Report1.SelectionFormula = RecSelect$
    Report1.action = 1
End Sub
```

If you are using a *date* in your record selection criteria, it might look like:

```
Sub Command1_Click ()
   'if we hard-code the date
   '(E.g. We are interested in reporting only records of March 04, 1993)
   RecSelect$ = "{file.field} = Date(1993,03,04)"
   MsgBox RecSelect$
  'The result of a message box should look exactly like the record selection if it
  'were built in Crystal Reports itself
   Report1.SelectionFormula = RecSelect$

   'if we use the value of a date/time variable of type Variant
   '(E.g. We are interested in reporting only today's records)
   TodayDate = Now
   RecSelect$ = "{file.field} = Date(" + Str$( Year (TodayDate)) + ","
   RecSelect$ = RecSelect$ + Str$( Month (TodayDate)) + ","
   RecSelect$ = RecSelect$ + Str$( Day (TodayDate)) + ")"
   MsgBox RecSelect$
  'The result of a message box should look exactly like the record selection if it
  'were built in Crystal Reports itself
```

```
  Report1.SelectionFormula = RecSelect$
  Report1.action = 1
End Sub
```

Note: you must manipulate the user inputted date within your VB code to fit the exact format of DATE(yyyy,mm,dd). If you try to pass any other format you will receive an "Error in Formula"

Formulas

Setting Formulas at Runtime

If you want to change a formula or a string (page title, author, etc...) at runtime from Visual Basic, you would take similar steps.

First, in Crystal Reports, create your formula. If you are giving it a string value, you must place a string value in it. This will force Crystal to treat the formula as a string. A single space inside double quotes is a good idea, as this will not print if left unchanged--yet it holds the string type for when you change the actual value later from VB. Place the new formula on your report where you would like the passed value displayed.

If you will be sending a numeric value or date, again place an appropriate value in the formula in order to establish the data type of the formula. To pass a numeric value to a formula, place 0 in the formula when you create the formula in Crystal Reports. To pass a date value to a formula, place the Today function in the formula.

Consider that you would like to send a title to your report from your VB application. The following is the suggested syntax to change the title at runtime. As instructed above, a formula is inserted at the top of the report, and is named *Title*. It presently contains a space inside quotes, as in " ".

```
Sub Command1_Click ()
  'if we hard-code the title
  Report1.Formulas(0) = "Title = 'Report on Telephone Usage'"
  'If we use the value of the Textbox
  Text1.Text = "Report on Telephone Usage"
  Report1.Formulas(0) = "Title = '" & Text1.Text & "'"
  Report1.action = 1
End Sub
```

Now you would like to send a numeric value to your report. The following code changes the value of a numeric formula at runtime.

```
Sub Command1_Click ()
  'if we hard-code the numeric value
  Report1.Formulas(1) = "Temperature = 65"
```

```
  'If we use the value of a variable
  new_temperature% = 65
  Report1.Formulas(1) = "Temperature = " & str$(new_temperature)
  Report1.action = 1
End Sub
```

Connect

```
Sub Command1_Click ()
'If you were to hard code the parameters at runtime
 Report1.Connect = "DSN=xxx;UID=yyy;PWD=****;DSQ=tttt"
'If you were to use variable name to store the user input for each parameter
 ConnectString$ = "DSN=" & Text1.Text & ";UID=" & Text3.Text & ";PWD=" &
Text4.Text & ";DSQ=" & Text2.Text
 Report1.Connect = ConnectString$

End Sub
```

Datafiles

```
Sub Command1_Click ()
'Set the Datafile location for each datafile in the report
  'under the Set Location in Crystal Reports dialogue a list
  'of files Is displayed and this list corresponds to the datafile number
  'Therefore, the first file listed would be 0, the second 1 etc.

  If LocationText.Text = "" Then
    MsgBox "You must enter a path to the datafile!"
    Exit Sub
  Else
    Main.Report1.DataFiles(DataFileNum.Text) = LocationText.Text
    Main!StatusBar.Caption = "Table Location " & DataFileNum.Text & " has been
set"
  End If

End Sub
```

UserName and Password

```
Sub Command1_Click ()
     Main.Report1.UserName = Text1.Text
     Main.Report1.Password = Text2.Text
End Sub
```

Sample Code:

```
Sub Command1_Click ()

'Set the Report file name

 Report1.ReportFileName = "C:\CRW4\SQL.RPT"

'Set Logon Parameters for SQL Rpt.

Report1.Connect = "DSN=CRSS;UID=sa;PWD=brahma;DSQ=pubs"

'Set the datafile locations which will override the
'existing file locations in the RPT file itself.
'once you set the datafiles once the next time you print
'the report the RPT will use the datafiles in the RPT again.
'Therefore, you must be sure to set the datafiles each time
'you want to print a report

 Report1.DataFiles(0) = "pubs.dbo.authors"

'Set a formula

Report1.Formulas(0) = "Parse = {authors.au_lname}[1 to 3]"

'Set selection formula based on formula

Report1.SelectionFormula = "{@Parse} = 'Whi'"

'Start the print job

Report1.Action = 1

End Sub
```