RIDCompX ActiveX Control

This unit is for the Report Interface Designer RIDCompX ActiveX control.

TRI-I Engineering Contact Information
License Agreement
Redistributable Files

Controls
RIDCompX

Types
TActionEvent
TErrorEvent

Constants
Error Constants used in the OnError event:

errBuildObjList	1099
errPopulateParams	1098
errPopulateSQLControls	1097
errShowRpt	1096
errRIDExecRpt	1095
errUnknownMajorVersion	1094
errUnknownMinorVersion	1093
errUnknownFileFormat	1092
errRIDBuildStream	1091

Action Constants used in the OnAction event:

actNothing	0
actOK	1
actCancel	2
actHelp	3
actClose	6
actLoad	11
actSave	12
actClear	13
actAutoLaunch	98
actInit	99

Status Constants used by the Status method:

stsBadDLL	-1
stsOk	0

http://www.iii-eng.com

RIDCompX Control

Properties Methods Events Tasks

Unit
RIDCompX

Description
RIDCompX is the Report Interface Designer "design-time"control for the ActiveX development environments. It provides a programmatic link between your developement tool, Report Interface Designer, and Seagate Crystal Reports.

Many of the properties and functionality can be tested at design-time. Set the DDFName property with an appropriate value then activate the Preview property. Preview will call the Execute method to display the RID dialog.

TRI-I Engineering Contact Information
License Agreement
Redistributable Files

http://www.iii-eng.com

Properties

- Run-time only
- Key properties
  - About
    - Database
    - DataSource
    - DDFName
    - Destination
    - DialogOnly
    - Name
    - Password
  - ParamStrs
    - Preview
    - RptDirectory
  - Tag
  - UserID

Methods

- Key methods
  - Execute
  - Status

Events

- Key events
  - OnAction
  - OnError

About the RIDCompX component

RIDCompX reference

Purpose

Use the RIDCompX control to connect your applications to the criteria dialogs you designed using the Report Interface Designer application.

Tasks

Add an instance of the RIDCompX control to your project by selecting from the palette and dropping onto your main form. In your code, set the DDFName property and then call the Execute method.

During the design phase, you can Preview what the RID dialog will look like, and as long as the DialogOnly property is FALSE, pressing the Ok button on the displayed dialog will activate Crystal Reports to display the report using the specified criteria.

About property

Applies to
RIDCompX control

Declaration
```
property About;
```

Description
Design-time pseudo property which displays an About Box for the RIDCompX control when the "..." ellipse button is pressed in the Object Inspector.

Database property

See also

Applies to
RIDCompX control

Declaration
```
property Database: String;
```

Description
This property is used in conjunction with the DataSource property. It will be populated when a report is assigned, but can be changed if desired. No verification is performed at the time of the change, so it is important that the new database has the same reporting structure.

See also

Datasource property
UserID property
Password property

DataSource property

See also

Applies to
RIDCompX control

Declaration
```
property DataSource: String;
```

Description
This property is a combobox listing all configured ODBC datasources. If the report is ODBC-based and the DSN for the report is configured on the current PC, then this value will be automatically selected. If the report is SQL-based, i.e., using a native driver, then the program will attempt to locate an ODBC DSN of that same name. If one cannot be found, the value will be set to *<unknown>*.

A valid ODBC configuration is only necessary if using either the "runtime-sourcing" capabilities of the ListBox or ComboBox controls. An interesting feature of both Crystal Reports and RID is that a combined native driver/ODBC driver combination can be implemented. If the report is based on a native driver but you would like to use the SQL controls, then it is necessary to configure an ODBC datasource targeted at the same server (and named the same) as the DataSource property of the TReport control in the RID DialogBox. In this way, data for the controls will be populated using an ODBC connection while the native connectivity will be used during the actual report execution.

See also

Database property
UserID property
Password property

DDFName property

See also

Applies to
RIDCompX control

Declaration
```
property DDFName: String;
```

Description
This is the name of the Dialog Definition File.

See also

RptDirectory property
Preview property
Execute method

Delimiter property

See also Example

Applies to
RIDCompX control

Declaration
```
property Delimiter: Char;
```

Description
<<< Description of Delimiter property >>>
Run-time only

See also

<<< See also of Delimiter property >>>

Delimiter property example

Destination property

Applies to
RIDCompX control

Declaration
```
property Destination: Integer;
```

Description
This property is not currently implemented.

DialogOnly property

See also Example

Applies to
RIDCompX control

Declaration
```
property DialogOnly: Boolean;
```

Description
This property determines if Crystal Reports will be called when the "Ok" button is pressed on the RID Dialog. The functionality may be useful if you are only interested in returning the selection criteria chosen by the user.

See also

Preview property
Execute method

DialogOnly property example

The following code is a modified copied from the RIDView sample program. This program has a combobox that stores the name of the files selected using an OpenDialog control. When a toolbar button is pressed, this retrieves the filename associated with the currently selected combobox entry and sets the DDFName property of theRIDCompX control. The DialogOnly property is then set to TRUE, and the Execute method is called to display the dialog and permit the user to enter criteria. When the user presses the Ok button, Crystal Reports is not activated, though the OnAction event is still triggered.

```
procedure TMainForm.ExecMenuClick(Sender: TObject);
begin
  if (cbxDDF.Text <> '') then
    with RIDComp1 do
      begin
        DDFName := TStr(cbxDDF.Items.Objects[cbxDDF.ItemIndex]).Str;
        DialogOnly := TRUE;
        Execute;
      end;
end;
```

Name property

Applies to
RIDCompX control

Declaration
```
property Name;
```

Description
The name of the control.

ParamStrs property

Example

Applies to
RIDCompX control

Declaration
```
property ParamStrs: Variant;
```

Description
This property permits run-time assignment or retrieval of Parameter values, in "Name=Value" format. These values can be referenced or changed during the OnAction() event execution phase.

This property is implemented as a Variant Array of BSTRs. Review the example to see how individual values are accessed.

Run-time only

ParamStrs property example

The following example demonstrates how to access the dialog parameters during the OnAction event. This code is copied from the **RIDViewVB** sample program.

```
Private Sub RIDCompX1_OnAction(ByVal Action As Integer, aModalResult As Integer)
Dim varOut As Variant
Dim Upper1, Lower1, Upper2, Lower2, I, Ind, CRPos As Integer
Dim strAct, WS1, WS2, CR As String
    If (cbParams.Value <> 1) Then
        Exit Sub
    End If
    ' DETERMINE THE ACTION TYPE
    Select Case Action
        Case actNothing
          strAct = "actNothing"
        Case actOK
          strAct = "actOK"
        Case actCancel
          strAct = "actCancel"
        Case actHelp
          strAct = "actHelp"
        Case actClose
          strAct = "actClose"
        Case actLoad
          strAct = "actLoad"
        Case actSave
          strAct = "actSave"
        Case actClear
          strAct = "actClear"
        Case actAutoLaunch
          strAct = "actAutoLaunch"
        Case actInit
          strAct = "actInit"
    End Select

    ' GET THE PARAMETERS TO POPULATE THE FRMPARAM.MMOPARAMS CONTROL
    varOut = RIDCompX1.ParamStr
    Upper1 = UBound(varOut, 1)
    Lower1 = LBound(varOut, 1)

    WS1 = ""
    For I = Lower1 To Upper1
        WS1 = WS1 + varOut(I) + Chr(13) + Chr(10)
    Next

    ' POPULATE THE MMOPARAMS CONTROL AND SHOW THE FORM
    frmParam.Hide
    frmParam.mmoParams.Text = WS1
    frmParam.lblAct.Caption = strAct
    frmParam.Show vbModal, Me
    WS2 = frmParam.mmoParams.Text
    Unload frmParam

    'POPULATE ANY CHANGES BACK TO THE RIDCOMPX OBJECT
    If (Trim(WS1) <> Trim(WS2)) Then
        Dim Strs() As String
        ' CYCLE THROUGH WS2 SPLITTING INTO INDIVIDUAL STRINGS
        I = 1
        Ind = 0
        ReDim Preserve Strs(Ind)
        CR = Chr(13) + Chr(10)
        CRPos = InStr(I, WS2, CR)
        While (CRPos <> 0)
          Let Strs(Ind) = Mid$(WS2, I, CRPos - I)
          I = CRPos + 2
          Ind = Ind + 1
          ReDim Preserve Strs(Ind)
```

```
            CRPos = InStr(I, CR, WS2)
        Wend
        Strs(Ind) = Mid$(WS2, I)
        Upper2 = UBound(Strs, 1)
        Lower2 = LBound(Strs, 1)
        ReDim varOut(Upper2)
        For I = Lower2 To Upper2
          varOut(I) = Strs(I)
        Next
        RIDCompX1.ParamStr = varOut
    End If
    Set varOut = Nothing
End Sub
```

ParentHWnd property

Applies to

RIDCompX control

Declaration

```
property ParentHWnd: HWND;
```

Description

This "write-only" property permits assignment of a Parent window for the Report output window. Use this when creating an MDI application where you wish to fully contain the report window within a child window.

Run-time only
Write-only

Password property

See also

Applies to
RIDCompX control

Declaration
```
property Password: String;
```

Description
This property is used in conjunction with the UserID property, and needs to be set when using any SQL controls (TComboBox, TListBox) associated with a protected database. This property will be populated when a Report is assigned.

See also

Database property
Datasourceproperty
UserID property

Preview property

See also

Applies to
RIDCompX control

Declaration
```
property Preview: String;
```

Description
This is not really a property, but a mechanism for viewing RID dialogs at "design-time". Set the DDFName property and then activate the ***Preview*** mode to see the actual RID dialog. If DialogOnly is set to TRUE, then Crystal Reports will not be executed.

See also

Execute method

RptDirectory property

See also

Applies to
RIDCompX control

Declaration
```
property RptDirectory: String;
```

Description
This property determines where RID will look for the Crystal Reports .RPT files. The report file will be located based on the following criteria:

1. If the *RptDirectory* property has a value, use it to find the .RPT file.
2. If the .RPT file is not found using the *RptDirectory* value, check in the same location as the DDF file.
3. If not found where the DDF file is, then use the path stored when the Report Association was made in the Report Interface Designer application.
4. If not found, then look in the directory where the current application was executed from.
5. If still not found, display an error.

See also

DDFName property

Tag property

Applies to
RIDCompX control

Declaration
```
property Tag;
```

Description
The tag property for the control.

UserID property

See also

Applies to
RIDCompX control

Declaration
```
property UserID: String;
```

Description
This property is used in conjunction with the Password property, and needs to be set when using any SQL controls (TComboBox, TListBox) associated with a protected database. This property will be populated when a Report is assigned.
Protected

See also

Database property
Datasource property
Password property

Execute method

See also Example

Applies to
RIDCompX control

Declaration
```
function Execute: Integer;
```

Description
Call this method to initiate the Dialog display and Report creation execution sequence.

See also

Preview property

Execute method example

The following code is copied from the **RIDView** sample program. This program has a combobox that stores the name of the files selected using an OpenDialog control. When a toolbar button is pressed, this retrieves the filename associated with the currently selected combobox entry and sets the DDFName property of theRIDCompX control. Then the Execute method is called to display the dialog and permit the user to enter criteria.

```
procedure TMainForm.ExecMenuClick(Sender: TObject);
begin
  if (cbxDDF.Text <> '') then
    with RIDComp1 do
      begin
        DDFName := TStr(cbxDDF.Items.Objects[cbxDDF.ItemIndex]).Str;
        Execute;
      end;
end;
```

Status method

Example

Applies to
RIDCompX control

Declaration
```
function Status: Integer;
```

Description
Call this method to check the status of the RID Engine. Result values are either stsOk (0) or stsBadDLL (-1), which usually occurs when theRIDCompX control cannot find the RIDExec.dll file.

Status method example

This code uses a button and a RIDCompX control on a form. When the user clicks the button, the code tests the Status of the control.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  StatRes : Integer;
begin
  StatRes := RIDComp1.Status;
  case StatRes of
    stsOk   : ShowMessage('Status is Ok');
    stsBadDLL        : ShowMessage('Status is BadDLL.');
  end;
end;
```

OnAction event

See also Example

Applies to
RIDCompX control

Declaration

Visual Basic 5.0
```
type TActionEvent =  procedure(ByVal Action As Integer, aModalResult As
                     Integer)

property OnAction: TActionEvent;
```

Description
This event is fired when a button is pressed on the RID dialog.

Action is one of the following values:

actNothing	This is a default value which should never occur.
actOK	Indicates that an "Ok" button was pressed.
actCancel	Indicates that a "Cancel" button was pressed.
actHelp	Indicates that a "Help" button was pressed.
actClose	Indicates that a "Close" button was pressed.
actLoad	Indicates that a "Load" button was pressed.
actSave	Indicates that a "Save" button was pressed.
actClear	Indicates that a "Clear" button was pressed.
actAutoLaunch	Indicates that the "AutoLaunch" property for the Report is TRUE. The OnAction event will be called with this value instead of the actInit value.
actInit	The event gets called with this value prior to the Dialog display.

aModalResult is the default ModalResult value for the pressed button. It can be changed at this time to keep the dialog on the screen.

The ParamStrs property is accessible during this event, and default control values can be set prior to the display of the dialog.

See also

OnError event

OnAction event example

The following example demonstrates how to overwrite the standard action handling to process the dialog parameters. It is the Action handling functionality of the **RIDView** sample program.

```
procedure TMainForm.RIDComp1Action(Action: Word; var aModalResult: Word);
var
  PL   : TStringList;
  Act : String;
  PF   : TParamForm;
begin
  if NOT(FParamsFlag) then
    Exit;
  try
    PL := TStringList.Create;
    PF := TParamForm.Create(Self);
    case Action of
      actNothing      : Act := 'actNothing';
      actOk           : Act := 'actOK';
      actCancel       : Act := 'actCancel';
      actHelp         : Act := 'actHelp';
      actClose        : Act := 'actClose';
      actLoad         : Act := 'actLoad';
      actSave         : Act := 'actSave';
      actClear        : Act := 'actClear';
      actAutoLaunch : Act := 'actAutoLaunch';
      actInit         : Act := 'actInit';
    end;
    with PF do
      begin
        PL.Assign(RIDComp1.ParamStrs);
        ParamStrs :=PL;
        CurrAction := Act;
        ShowModal;
        PL.Text := ParamStrs.Text;
        RIDComp1.ParamStrs := PL;
      end;
   finally
    PF.Free;
    PL.Free;
  end;
end;
```

OnError event

See also Example

Applies to
RIDCompX control

Declaration

Visual Basic 5.0
```
type TErrorEvent =  procedure(ByVal ErrorNum As Integer, ByVal ErrorText As
                    String, Handled As Boolean)

property OnError: TErrorEvent;
```

Description
This event is fired when an error is detected.

ErrorNum and *ErrorText* are as follows:

errBuildObjList	1099	*Depends on where error occurs*.
errPopulateParams	1098	*Depends on where error occurs*.
errPopulateSQLControls	1097	*Depends on where error occurs*.
errShowRpt	1096	*Depends on where error occurs*.
errRIDExecRpt	1095	*Depends on where error occurs*.
errUnknownMajorVersion	1094	Unknown Major Version in RID File.
errUnknownMinorVersion	1093	Unknown Minor Version in RID File.
errUnknownFileFormat	1092	Unknown File Format.
errRIDBuildStream	1091	*Depends on where error occurs*.

Handled is a boolean value with a default of FALSE. Setting this value to TRUE will stop RID from displaying the error message.

See also

OnAction event

OnError event example

The following example, copied from the **RIDView** sample application, demonstrates how to overwrite the standard error handling to display custom errors.

```
procedure TForm1.RIDComp1Error(ErrorNum: Word; ErrorText: PChar;
  var Handled: Boolean);
begin
  Handled := True;
  case ErrorNum of
    errBuildObjList :
      begin
        MessageBox(0, PChar(Format('[%d] "BuildObjList" Error: [%s]',
            [ErrorNum, ErrorText])),
          'Report Interface Designer', MB_ICONEXCLAMATION or MB_OK);
      end;
    errPopulateParams :
      begin
        MessageBox(0, PChar(Format('[%d] "PopulateParams" Error: [%s]',
            [ErrorNum, ErrorText])),
          'Report Interface Designer', MB_ICONEXCLAMATION or MB_OK);
      end;
    errPopulateSQLControls :
      begin
        MessageBox(0, PChar(Format('[%d] "PopulateSQLControls" Error: [%s]',
            [ErrorNum, ErrorText])),
          'Report Interface Designer', MB_ICONEXCLAMATION or MB_OK);
      end;
    errShowRpt :
      begin
        MessageBox(0, PChar(Format('[%d] "ShowRpt" Error: [%s]',
            [ErrorNum, ErrorText])),
          'Report Interface Designer', MB_ICONEXCLAMATION or MB_OK);
      end;
    errRIDExecRpt :
      begin
        MessageBox(0, PChar(Format('[%d] "RIDExecRpt" Error: [%s]',
            [ErrorNum, ErrorText])),
          'Report Interface Designer', MB_ICONEXCLAMATION or MB_OK);
      end;
  end;
end;
```

TActionEvent type

See also

Unit
RIDCompX

Declaration
```
type TActionEvent =  procedure(Action: Word; var aModalResult: Word) of
                     Object;
```

Description
Assign a handler to this event to respond to a button-press action. The Action value corresponds to one of the act???? constants. The aModalResult value corresponds to the default value assigned to the pressed button. You can change this value to modify the default behavior.

See also

OnAction event

TErrorEvent type

See also

Unit
RIDCompX

Declaration
```
type TErrorEvent =  procedure(ErrorNum: Word; ErrorText: PChar; var Handled:
                    Boolean) of object;
```

Description
Assign a handler to this event to trap any errors that are generated during the Execute() phase. The ErrorNum value corresponds to one of the err???? constants. The ErrorText value is the default error message. Handled defaults to FALSE. If you set it to TRUE, the RID engine will not display an error message.

See also

OnError event

Contacting TRI-I Engineering

We provide free Technical Support to registered users for 60 days from purchase or registration. To receive support, you will be asked to provide your product serial number. Technical Support for the Evaluation version is not available, however, we will attempt to answer via e-mail any support questions you may have during the evaluation period.

Address:

TRI-I Engineering, Inc.
200 W. 17th Street, Suite 80
Cheyenne, WY 82001

Telephone:

Sales & Information
(888) 551-3500

Support
(310) 967-3966

World Wide Web:

http://www.iii-eng.com

sales@iii-eng.com
support@iii-eng.com

License Agreement

TRI-I Engineering, Inc. authorizes you to make archival copies of this software for the sole purpose of back-up and protecting your investment from loss. Under no circumstances may you copy this software or documentation for the purposes of distribution. You are not to remove any of the copyright notices included in the software or product documentation.

You may distribute, without runtime fees or further licenses, your own compiled programs based on any of the compiled units and/or components included with Report Interface Designer (RID). You may also distribute, without runtime fees or further licenses, the RIDEXEC.DLL dynamic link library, as well as any compiled sample programs contained with this package. You may not distribute any of the Report Interface Designer (RID) design-time components or the designer application without written permission from TRI-I Engineering, Inc.

The previous restrictions do not prohibit you from distributing your own source code, units, or components that depend upon Report Interface Designer (RID). However, others who receive your source code, units, or components need to purchase their own copies of Report Interface Designer (RID) in order to compile the source code or to write programs that use your units or components which are dependent on either the Report Interface Designer (RID) components or the RIDEXEC.DLL dynamic link library.

The supplied software may be used by one person on as many computer systems as that person uses. Group programming projects making use of this software must purchase a copy of the software and documentation for each member of the group.

TRI-I Engineering, Inc. warrants that the physical CD or diskettes and documentation provided with Report Interface Designer (RID) shall be free of defects in materials and workmanship for a period of 60 days from the date of receipt. If you notify us of such a defect within the warranty period, TRI-I Engineering, Inc. will replace the defective CD, diskette(s), or documentation at no cost to you.

TRI-I Engineering, Inc. warrants that the software will function as described in this documentation for a period of 60 days from receipt. If you encounter a bug or deficiency, we will require a problem report detailed enough to allow us to find and fix the problem. If you properly notify us of such a software problem within the warranty period, TRI-I Engineering, Inc. will update the defective software at no cost to you.

TRI-I Engineering, Inc. further warrants that the purchaser will remain fully satisfied with the product for a period of 60 days from receipt. If you are dissatisfied for any reason, and TRI-I Engineering, Inc. cannot correct the problem, contact the party from whom the software was purchased for a return authorization. If you purchased the product directly from TRI-I Engineering, Inc., we will refund the full purchase price of the software (not including shipping costs) upon receipt of the original program CD or diskette(s) and documentation in undamaged condition. TRI-I Engineering, Inc. cannot offer refunds directly to anyone who did not purchase a product directly from us.

Disclaimer of Warranty
TRI-I ENGINEERING, INC. DOES NOT ASSUME ANY LIABILITY FOR THE USE OF REPORT INTERFACE DESIGNER (RID) BEYOND THE ORIGINAL PURCHASE PRICE OF THE SOFTWARE. IN NO EVENT WILL TRI-I ENGINEERING, INC. BE LIABLE TO YOU FOR ADDITIONAL DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF OR INABILITY TO USE THESE PROGRAMS, EVEN IF TRI-

I ENGINEERING, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Governing Law
This agreement shall be governed by the laws of the State of Wyoming, United States of America.

Redistributable Files

Addendum to the TRI-I Engineering, Inc. License Agreement

TRI-I Engineering, Inc. grants you a royalty-free license to distribute applications based on the RIDComp and RIDCompX
components and libraries. Following is the list of redistributable files:

- RIDExec.dll — Every client will need this file installed in either the Application directory (preferred) or in the Windows System directory.

- RIDCompX.ocx, RIDCompX.tlb — These files are only required if you are developing with the RID ActiveX component. As part of your own installation program, this file will need to be copied to the client Windows System directory and then registered. If your installation program does not automatically detect and register OCX files, then you will need to manually register this component using REGSVR32.EXE. Following is an example:

  REGSVR32 <ocx directory>\RIDCompX.ocx

In no case may you redistribute with your application any of the design-time components, including but not limited to RIDCOMP.DCU and III.DPL, any ActiveX control license (.LIC) files, or the Report Interface Designer application (RID.EXE) itself.