

Crystal Reports

Getting Started with the Crystal Reports VCL in Borland Delphi 5

Overview

This document discusses using the Crystal Reports Visual Component Library (VCL) with Borland Delphi 5.

Examples of common report handling tasks, such as previewing, logging on to a database, and passing parameters are presented.

This document is for use with Crystal Reports 7 and higher.

Contents

INTRODUCTION	1
INSTALLING THE VCL	2
WORKING WITH THE VCL.....	4
CONNECTING TO A DATABASE	5
<i>LogonServer</i>	5
<i>LogOnInfo</i>	6
<i>Connect</i>	7
PASSING AN ACTIVE DATA RECORDSET TO A REPORT	8
PASSING PARAMETER VALUES	9
WORKING WITH A SUBREPORT	9
FINDING MORE INFORMATION.....	10
CONTACTING CRYSTAL DECISIONS FOR TECHNICAL SUPPORT	10

Introduction

The VCL is a wrapper around the Crystal Reports Print Engine (crpe32.dll) and uses direct Windows API calls to access this engine. The source code for the VCL is included in the ucrpe32.pas file. The direct API calls are defined in the crDynamic.pas file.

The VCL you will use with this document is designed for Crystal Reports 7, but is compatible with versions 5, 6 and 8.

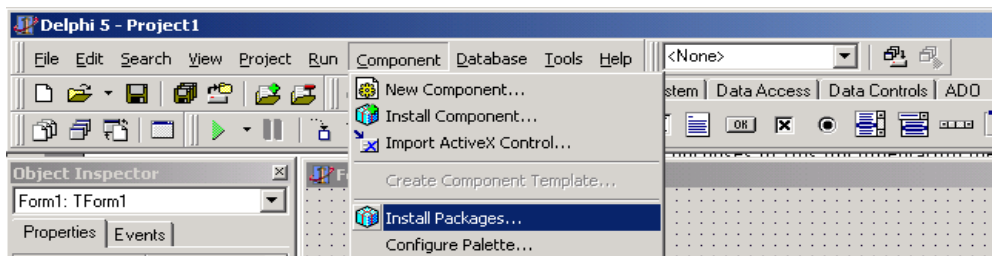
Installing the VCL

To install the VCL:

1. Extract the Borland Delphi 5 VCL for Crystal Reports (from 7vcl551.exe) to a directory of your choice. For purposes of this document, the VCL is extracted to 'C:\Program Files\Borland\Delphi5\7vcl551'.

NOTE	To obtain the Borland Delphi 5 VCL for Crystal Reports: • Search and download 7vcl551.exe from http://support.crystaldecisions.com/downloads. or • Copy 7vcl551.exe from the Crystal Reports 8 Developer Edition CD.
-------------	--

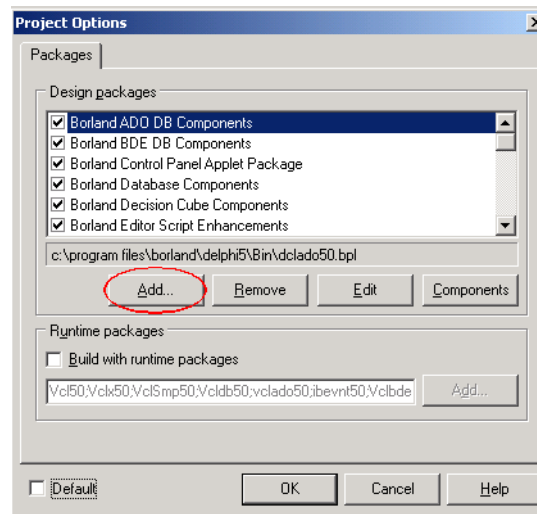
2. Open Borland Delphi 5 and click on the **Component** menu, then select **Install Packages**.



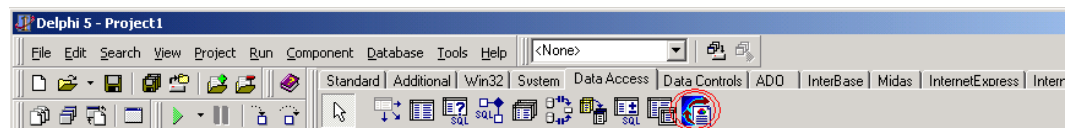
3. The **Project Options** dialog box appears. Click on the **Add** button then browse to 'C:\Program Files\Borland\Delphi5\7vcl551' and choose the **Crystal.bpl**.

This type of file is referred to as a Borland Package Library. This package library is the easiest way to install the VCL into Delphi.

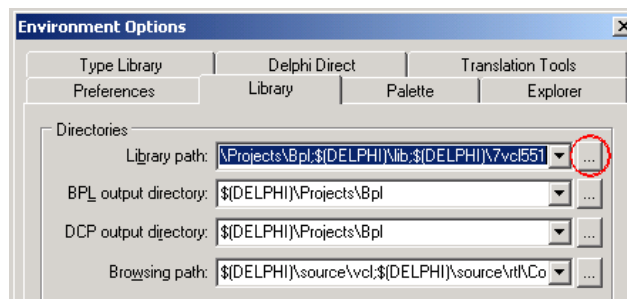
Under the Design Packages area, an entry for the **Crystal Reports VCL for Delphi 5** appears. Click **OK**.



Under the Data Access Tab, the VCL (with a type of Terpe) appears.



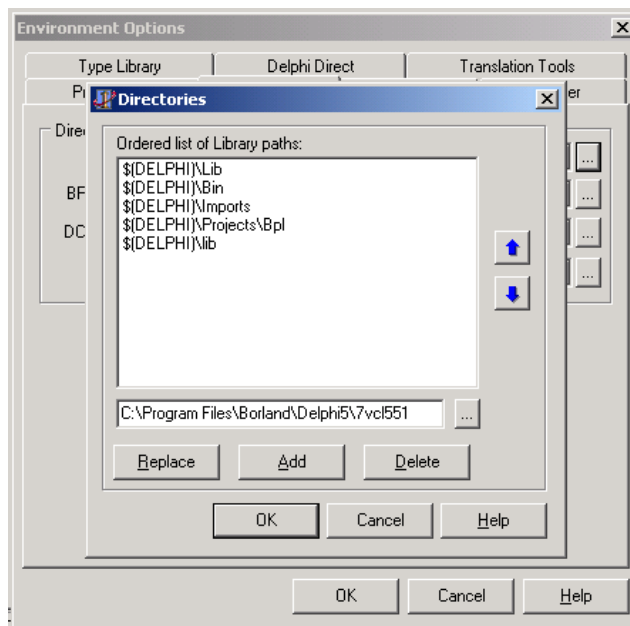
4. Go to the **Tools** menu and select **Environment Options**. The **Environment Options** dialog box appears. Go to the **Library** tab and click the ellipse button to the right of the **Library path** combo box.



5. The **Directories** dialog box appears. Using the ellipse button, browse to 'C:\Program Files\Borland\Delphi5\7vcl551' or type the path into the text box to the left of the ellipse button.

Click on the **Add** button, then on the **Ok** button.

The VCL is now installed.



Working with the VCL

When the VCL is added to a Delphi project its default name is **Crpe1** and is of type **TCrpe**. To add the VCL to a new project, go to the **Data Access** tab and double click the VCL. In addition, add a button from the **Standard** tab.

On your form, you should now have the VCL and a button.

To preview a report (that uses a native connection to a PC type database), double click the button and type the following code:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Crpe1.ReportName := 'C:\<some path>\your report.rpt';
    Crpe1.Execute;
end;
```

NOTE

The **ReportName** property is the equivalent to the **ReportFileName** property of the Crystal ActiveX (OCX) control.

The **Execute** property is the equivalent to the **Action** property of the Crystal ActiveX (OCX) control.

Delphi uses single quotes to enclose a string

To run the application, press **F9** or go the **Run** menu and select **Run**. Press the button to preview the report.

Connecting to a Database

To connect to a database, there are three methods that you can use:

- **LogonServer**
- **LogonInfo**
- **Connect**

LogonServer

The LogOnServer object is a run-time only object that contains properties that allow for logging onto a server. The difference between this object and the Connect and LogOnInfo objects is that the latter work on a table-by-table basis, setting the log on information for each table in a Report. However, LogOnServer just opens a connection to a SQL Server and if the Report has that server specified in it, it will use the connection.

The advantage to LogOnServer is that it is simple, and only needs to be opened once for any reports (and Subreports) that use that particular Server, whereas the Connect and LogOnInfo properties need to be specified on a report-by-report basis.

The disadvantage is that it will not change the Server name that is specified in a Report, therefore it is not possible to use LogOnServer if the Server name is going to be different than when the report was created.

The parameters for the **LogOnServer** property are:

ServerName (When using ODBC, use the data source name (DSN) instead)
UserID
Password
DatabaseName
DLLName

```
// Sample code of using LogonServer
// Create a LogOnServer item
Crpel.LogOnServer.Add;
// Populate the logon information
Crpel.LogOnServer.DLLName := 'P2SSQL.DLL';
Crpel.LogOnServer.ServerName := 'DBConn1';
Crpel.LogOnServer.UserID := 'vantech';
Crpel.LogOnServer.Password := 'vantech';
Crpel.LogOnServer.DatabaseName := 'pubs';
// Test the logon information
if Crpel.LogOnServer.LogOn then
    ShowMessage('The LogOn ID is: ' + IntToStr(
Crpel.LogOnServer.Number))
else
```

```

        ShowMessage('Error Logging on to Server' +
        IntToStr(Crpe1.LastErrorNumber));
    end;

```

NOTE	- To find the DLLName for a report, in the Crystal Designer go to the Database menu and select Convert Database Driver. The DLLName is the value displayed beside the From field. - To find the Server, Database, UserID parameters for the LogonServer method, go to the Database menu and select Set Location. The dialog screen that appears will contain the members just mentioned.
-------------	---

LogOnInfo

The LogOnInfo object is used to set the log on information for each table in the report. It applies to reports based on SQL tables or ODBC data sources, as does the Connect property, and LogOnServer.

The difference between LogOnInfo and Connect is that LogOnInfo allows different log on parameters for each table, whereas Connect takes one set of parameters and applies them to all the tables in a report. Therefore, if a report is designed with tables from more than one Server, LogOnInfo should be used instead of Connect.

The parameters for LogOnInfo are:

ServerName (When using ODBC, use the data source name (DSN) instead)
 UserID
 Password
 DatabaseName

NOTE	Unlike LogOnServer, there is no need to pass the DLLName argument (the DLL is read from the report).
-------------	--

```

// Sample code of LogOnInfo
var
    cnt : integer;
begin
    // Tell the VCL to use the LogonInfo methods to make
    // a connection
    Crpe1.ConnectMethod := useLogOnInfo;

    // Retrieve the logon information from the report
    Crpe1.LogOnInfo.Retrieve;
    // Loop through each of the tables in the report
    for cnt := 0 to (Crpe1.LogOnInfo.Count - 1) do
    begin

```

```

Crpel.LogOnInfo[cnt];
//Populate the Logon Information
Crpel.LogOnInfo.ServerName := 'DBConn1';
Crpel.LogOnInfo.UserID := 'vantech';
Crpel.LogOnInfo.Password := 'vantech';
Crpel.LogOnInfo.DatabaseName := 'pubs';
//Test the Logon Information
if Crpel.LogOnInfo.Test then
    ShowMessage('LogOn information is good')
else
    ShowMessage('Error with LogOn information');
end;

```

NOTE

- To find the Server, Database, UserID parameters for the LogonInfo method, go to the **Database** menu and select **Set Location**. The dialog screen that appears will contain the members just mentioned.

Connect

Whereas LogOnInfo allows different LogOn parameters for each table, Connect takes one set of parameters and applies them to all the tables in a report. Therefore, if a report is designed with tables from more than one server, LogOnInfo should be used instead of Connect.

However, if the main report uses one server, and the Subreport uses another, then Connect can still be used, since it can apply to Subreports as well.

The parameters for Connect are:

ServerName (When using ODBC, use the data source name (DSN) instead)
 UserID
 Password
 DatabaseName

NOTE

Unlike LogOnServer, there is no need to pass the DLLName argument (the DLL is read from the report).

```

// Example code of Connect

Crpel.ConnectMethod := useConnect;
// Retrieve the Connection information from the report
Crpel.Connect.Retrieve;
// Pass the connection information to the VCL
Crpel.Connect.ServerName := 'DBConn1';
Crpel.Connect.UserID := 'vantech';
Crpel.Conenct.Password := 'vantech';

```

```

Crpel.Connect.DatabaseName := 'pubs';
// Test the connection information
if Crpel.Connect.Test then
    ShowMessage('LogOn information is good')
else
    ShowMessage('Error with LogOn information');
end;

```

NOTE

Connection information should be passed to each subreport, see below for how to redirect the VCL to pass information to a subreport. If a report has tables from multiple data sources that require different logon information the use LogOnInfo should be used instead.

Passing An Active Data RecordSet to a Report

If a report has been designed to report off of an Active Data Recordset (such as ADO, RDO, DAO or CDO), use the **DataPointer** method of the **table** object.

The **DataPointer** property, along with the **Bytes** and **Tag** properties are part of the ADO/CDO support that has been added to the Tables class with the 7.x.0.20 release of the Crystal Reports VCL. The DataPointer should point to the address of a valid Data Object RecordSet, such as an Active Data Object (ADO) RecordSet, or the Crystal Data Object (CDO) RecordSet.

To pass an ADO Recordset to a report:

1. Add a TAdoQuery and a TAdoConnection from the ADO tab to form1.
2. Populate your Delphi TADOQuery with data as follows:

```

AdoConnection1.ConnectionString := 'Provider=MSDASQL;Data
Source=Xtreme Sample Database';
AdoConnection1.LoginPrompt := false;
AdoQuery1.Connection := AdoConnection1;
AdoQuery1.SQL.Clear;
AdoQuery1.SQL.Add ('select * from customer');
AdoQuery1.CursorLocation := clUseClient;
AdoQuery1.CursorType := ctStatic;
AdoQuery1.Open;

```

3. Pass that AdoQuery to the report:

```

Crpel.Tables.Retrieve;
Crpel.Tables[0].DataPointer := @(AdoQuery1.Recordset);

```

It is recommended to use only one table in a report when working with Active Data as linking/filtering/sorting is more efficiently done by Delphi's AdoQuery component than by Crystal Reports Active Data Driver (p2smon.dll).

For further information on using ADO in your Delphi application, refer to the Crystal Reports VCL help file (Ucrpe32.hlp) and search for **ADO** or **DataPointers**.

Passing Parameter Values

When reports have parameters, you can handle parameter value assignment either in a custom dialog or programmatically. To pass a parameter value to a report, use the ParamFields collection of the VCL.

The following examples demonstrate how to pass a date parameter and a string parameter to a report.

```
crpel.ReportName := 'C:\<some path>\your report.rpt';  
//Retrieve the parameters from the report  
crpel.ParamFields.Retrieve;  
//The first parameter in the report is a date  
crpel.ParamFields[0].Value := '1999, 02, 25';  
//The second parameter in the report is a string  
crpel.ParamFields[1].Value := 'Sample String';  
crpel.Execute;
```

Working with a Subreport

Many of the properties and sub-class objects on the VCL can apply to the main report, or to any Subreport as well. When Subreports are involved, the Subreport object is used to distinguish which report the VCL properties are currently pointing to.

The first item in the Subreports object (Subreports[0]) is always a reference to the main Report. To reference the first subreport, you would use Subreports[1].

For example, to change a formula field in a subreport use the following code:

```
//Set the Report Name  
  
crpel.ReportName := 'C:\<some path>\your report.rpt';  
// Allow the preview window to drill down into  
// the subreport  
  
crpel.WindowButtonBar.AllowDrillDown := true;  
  
crpel.Subreports.Retrieve;  
  
// Redirect the VCL to pass values to the first  
// subreport  
  
crpel.Subreports[1];  
  
crpel.Formulas.Retrieve;
```

```
//Choose a formula from that subreport to modify  
crpe1.Formulas.Name := 'header';  
  
// Clear the original value for the formula  
crpe1.Formulas.Formula.Clear;  
  
// Pass the new content to your formula  
crpe1.Formulas.Formula.Add('{@Month}+" My Changed  
Sales"');  
  
//point the VCL back to the main report  
crpe1.Subreports[0];  
  
crpe1.Execute;
```

Finding More Information

- CrystalVCL.dpr – Sample Delphi and VCL application. This sample comes with 7vcl551.exe.
- Ucrpe32.hlp – VCL Help file. This file is installed with Crystal Reports.
- VCL_Overview.exe – Using the VCL control for Crystal Reports 7. Download this file from <http://support.crystaldecisions.com/docs>.
- Apps_Delphi.pdf – Listing of Delphi Sample Applications. Download this file from <http://support.crystaldecisions.com/docs>.

Contacting Crystal Decisions for Technical Support

Along with this document, and the *Crystal Reports Developer's Help file*, we recommend that you visit our Technical Support web site for further resources and sample files. For further assistance, visit us at the websites below.

Technical Support web site:

<http://support.crystaldecisions.com/homepage/>

Answers By Email Support:

<http://support.crystaldecisions.com/support/answers.asp>

Phone Support:

Tel: (604) 669-8379