crystal decisions™

Crystal Reports for Visual Studio .NET

Crystal Reports Guide To ADO.NET

Overview

This document describes how to use Crystal Reports for Visual Studio .NET with ADO.NET. This document also covers the differences between ADO and ADO.NET

Contents

Introduction

Crystal Reports for Visual Studio .NET supports reports that access ADO.NET datasets.

You can create an ADO.NET dataset from a variety of sources. Whatever the source is, before you can report off an ADO.NET dataset, you must first generate an object for the dataset. Subsequently, you can use a Report Expert in Crystal Reports for Visual Studio .NET to create a new report based on the data description provided by the dataset object.

A dataset object generated with the Visual Studio ADO.NET Dataset Designer contains only the data description, and not the actual data. Consequently, when working with a report connected to such a dataset object, you cannot browse field data in the Crystal Report Designer at design time.

To have the report display the actual data at runtime, you must first push the data into the dataset object using the ADO.NET Object Model, and then pass the populated dataset to the Report Engine using the Report Engine Object Model. You must then bind the report to the corresponding Web or Windows Forms Viewer before building and running the application.

NOTE	The samples in this document are based on creating a Windows application based on an ADO.NET dataset in Visual Basic .NET using Crystal Reports 9.

Differences Between ADO and ADO.NET

ADO

When using ADO, three main objects are used to access and manipulate data:

- The connection object
- The command object
- The recordset object

When creating a recordset from a database connection, two types of cursors may be used:

- The server side cursor - This cursor allows the recordset to be generated on the database server. The recordset works with the data from the server directly. No data is passed to the client machine.
- The client-side cursor – This cursor can also be used to generate the recordset. In this case, the data is passed down to the client machine where it can be manipulated.

It is possible to generate data in a recordset without having any connection to a database. This is accomplished through SHAPE commands.

NOTE	For further information on SHAPE commands, refer to Microsoft's Developer Website at http://msdn.Microsoft.com

One disadvantage when using ADO is that it is difficult to pass large COM objects around a network. It is also very difficult to share recordsets across computing platforms and recordsets cannot penetrate firewalls.

ADO.NET

When using ADO.NET, three main objects are used to access and manipulate data:

- The dataset object – This object is a disconnected recordset object. Use this object with Crystal Reports.
- The datacommand object – Use this object to pass a query to a database to generate records
- The datareader object – This server-side forward only object is used for reading data to a table or a list box

Dataset object

Unlike a recordset, the **dataset object** is completely separate from the database. In fact, the **dataset object** knows nothing about the database. With ADO.NET, the connection is always client-side and does not face the same limitations when updating the server with information.

In all cases, we can update the database using ADO.NET. Datasets also allow data shaping without having to use the SHAPE command from ADO.

The most powerful feature of ADO.NET is its integration with Extensible Markup Language (XML). You could consider datasets the result of bringing together ADO and XML. A dataset contains one or more lumps of tabular XML, known as **datatables**. Unlike ADO, ADO.NET is able to obtain information from each row in the datatable. It is very easy to take a dataset and convert it to an XML document. You can also take an XML document and convert it into a dataset.

There are also no issues passing the data across a network due to the fact that the dataset is not a large COM object. It is simply an XML stream.

How Crystal Reports Works With ADO and ADO.NET

RDC 8.5 and ADO

Before ADO.NET, it was possible to create a report that used ADO through the Report Designer Component (RDC). The recordset would be passed to the report at runtime. In order to this to work successfully certain conditions needed to be met.

First of all, the report would need to be created off a field definition file (or TTX file). This tab separated text file describes the structure of the recordset to the report. This file could be created manually or it could be created off of a function, called **CreateFieldDefFile**, built into the Crystal Active Data database DLL, **P2SMON.DLL**.

Once connected to this TTX file, the report would treat th TTX file as an ordinary table. Fields from this table could be placed onto the report.

Once the report was saved, the TTX file could be discarded. You would only need the definition file again if the recordset structure changes.

At runtime use the **SetDataSource** function to pass this recordset to the report.

For more information on using the 8.5 RDC with ADO, download **'scr8_ttxado.pdf'** from:

http://support.crystaldecisions.com/docs

RDC 8.5 and XML

The Report Designer Component (RDC) 8.5 supports XML, although using XML at runtime is rather limited.

In order for a report to connect to an XML data source, an ODBC connection has to be created to point to a physical XML file. You can then build a report off this ODBC DSN.

This method works successfully as long as the XML file structure does not change and the ODBC DSN is created successfully.

When distributing a report that uses XML, you must distribute the ODBC drivers with the application. Some registry entries would have to be made and the DSN would have to be setup on the client. Using the RDC, it is very difficult to change the XML file you used at runtime.

For more information on using the RDC 8.5 with XML, download **'cr_xml_data_sources.pdf"** from:

http://support.crystaldecisions.com/docs

RDC 8.5 and ADO.NET

When the Report Designer Component (RDC) 8.5 was created, datasets did not exist. Although the RDC will work with traditional ADO it will not work with ADO.NET dataset files.

The only way to use ADO.NET with Crystal would be to use Crystal Reports for Visual Studio .NET or Crystal Reports 9 Developer or Advanced Server.

Crystal Reports for VS .NET and ADO

You can use Crystal Reports for Visual Studio .NET with traditional ADO recordsets. The code used to create the recordset would be identical to the code you would use when working with the Report Designer Component (RDC) component.

Use the same method (**SetDatasource)** that the RDC uses to pass the recordset to the report, however, in this case you would pass the recordset to the Report Document object.

Using this connection type includes all the limitations described earlier in the RDC 8.5 and ADO section of this document.

For a sample application using Crystal Reports for VS .NET and ADO, download **'vbnet_win_classicado.exe'** from:

http://support.crystaldecisions.com/downloads

Crystal Reports for VS .NET and ADO.NET

Since Crystal Reports for Visual Studio .NET was designed with Microsoft Visual Studio .NET (VS .NET), you can take advantage of all the new features of VS.NET including the ability to design your reports off ADO.NET datasets.

Working with VS .NET and ADO.NET datasets is similar to using the RDC and TTX files. When designing a report in VS .NET, rather then connecting the report to a TTX file, connect the report to a dataset (.XSD) file.

Create the XSD file by:

- Using a XML stream
- Saving a dataset to file
- Creating a dataset using the designer within the .NET environment

Once the report points to the .XSD file, the report reads the structure of the file to see what fields are available to add to the report.

The report can then be designed based on these exposed fields. Once the report has been created, pass the data to the report at runtime. This is done using the **SetDataSource** method.

When you pass the dataset, the report knows exactly where to place the fields as the report knows the structure of the data being passed.

At this point, you will still run into the same limitation that the RDC 8.5 had with TTX files. If the structure of the dataset changes at runtime and it is passed to the report, the report will no longer know where to place these new fields. Most commonly, this results in invalid or incorrect data on the report.

It is very important to keep the structure of the dataset consistent with the structure of the XSD file the report uses.

The best feature of using datasets over the more traditional database connection methods is that you can use your dataset for the entire application. This one set of data can be displayed to a grid where the user may view and filter the data.

When the user is ready to print the report, all you need to do is pass this filtered dataset to the report. The report does not need to make any connections to the database. This method is much faster when compared to a database connection.

Deploying an application

For information on deploying an application that uses Crystal Reports for Visual Studio .NET and ADO.NET, download **'crnet_deployment.pdf'** from:

http://support.crystaldecisions.com/docs

ADO.NET Tutorials

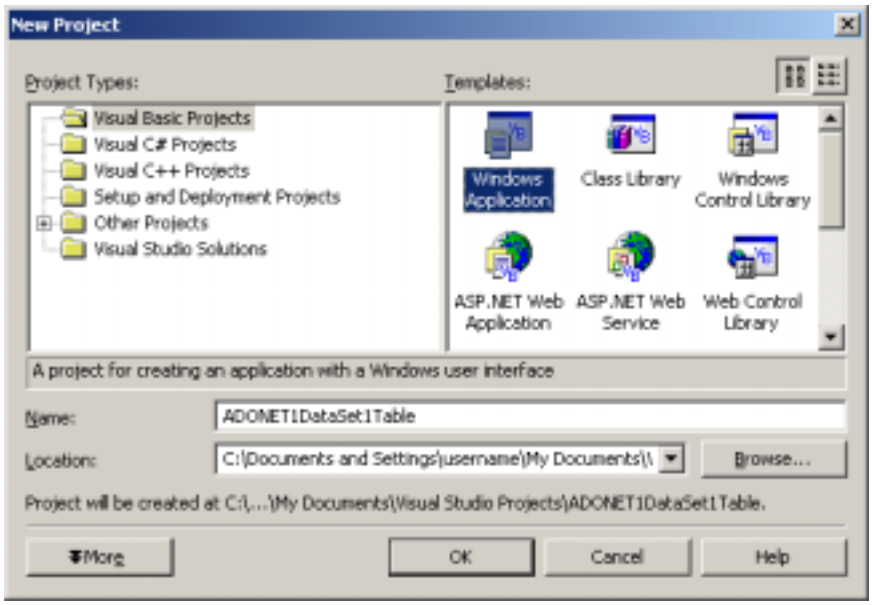
Tutorial 1 - Using a Single DataSet and Table

This tutorial explains how to create a report using Crystal Reports for Visual Studio .NET and ADO.NET. This tutorial describes creating a new report based on a dataset file and passing the dataset to the report.

1. Start Visual Studio .NET and select **New Project** from the Visual Studio .NET start page. The **New Project** dialog box appears.

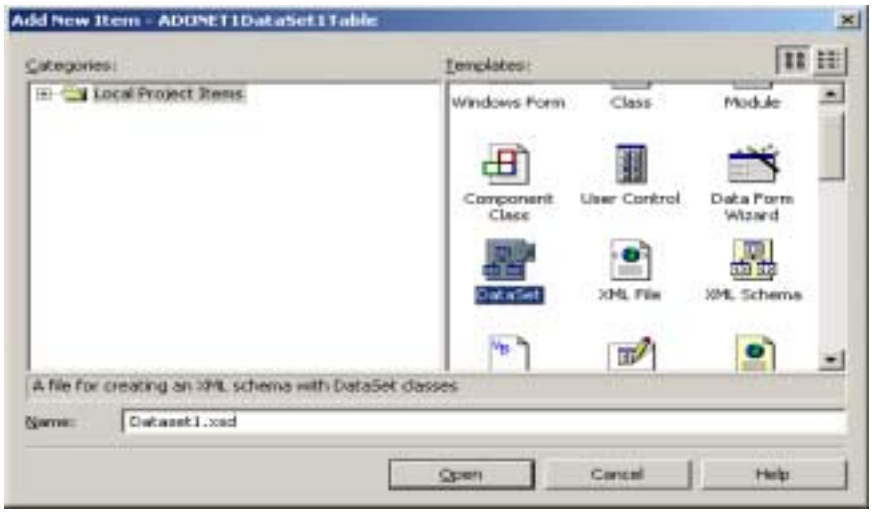
2. Select **Visual Basic Projects** as the Project Type and select **Windows Application** as the Template.

   Name the application, **ADONET1DataSet1Table**. Select **OK**. You should now see a VB form.



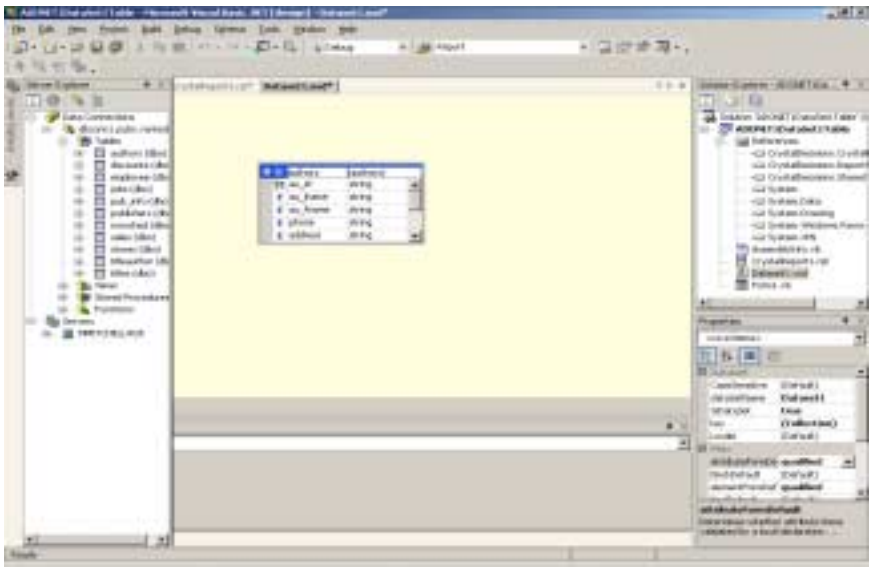
Add a dataset

1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **dataset** object and keep the default name, **Dataset1.xsd.** Select **Open**.



2. Once the dataset has been added to the project, browse through the **Server Explorer** to find a table to add to this dataset. In this example, find the Microsoft SQL Server sample database, **Pubs** and locate the **authors** table.

   Drag the **authors** tables from the **Server Explorer** to the **dataset** tab of Dataset1.xsd.



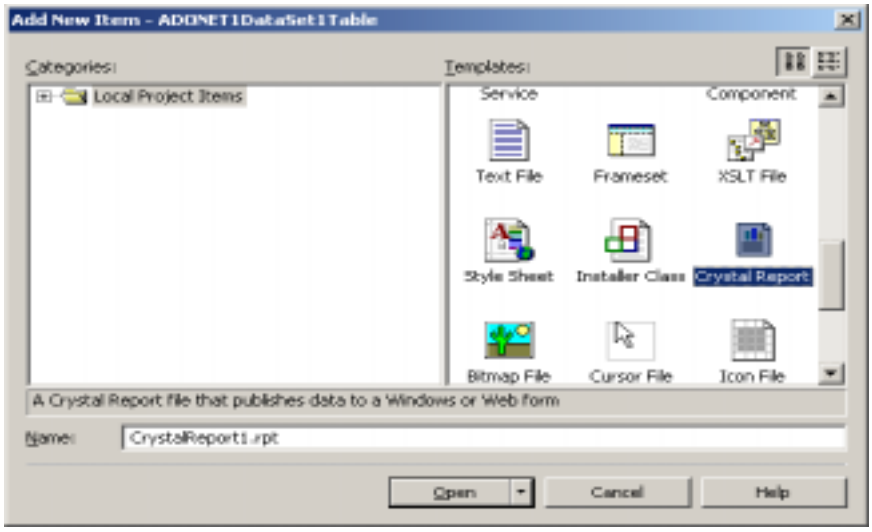
3. On the **Schema** menu, click **Generate Dataset** to generate the dataset object for the project. In this scenario, the dataset object will contain the data description for the **authors** table.

   The generated dataset object contains only a description of the database.

4. Save your dataset (dataset1.xsd).

Build a Report

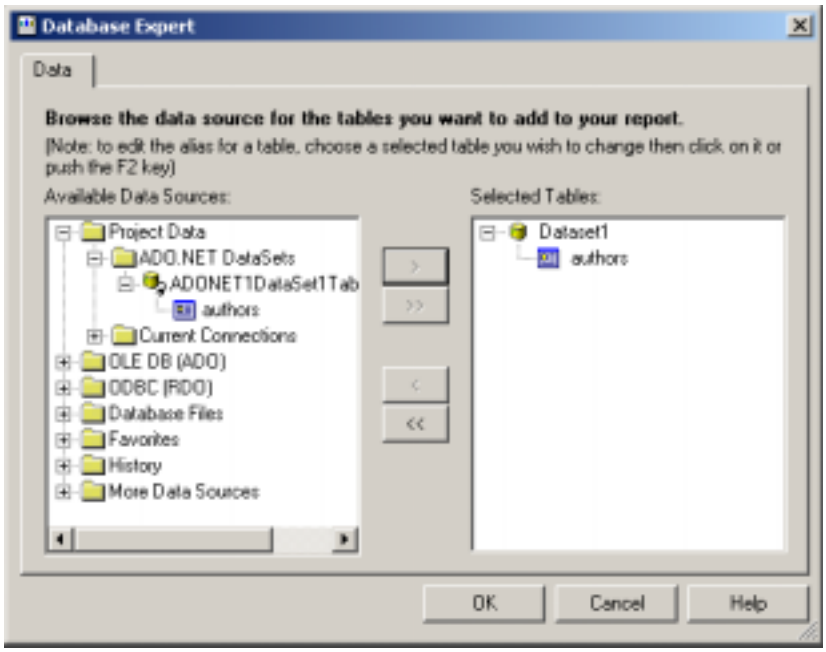
1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **Crystal Reports** template and keep the default name, **CrystalReports1.rpt**. Click **Open**.



2. The **Crystal Report Gallery** appears. Select the **As Blank Report** option and click **OK**. The design view of the report appears including the **Field Explorer** on the left hand side.

3. In **the Field Explorer**, right-click **Database Fields** option and select **Database Expert.**

   The **Database Expert** dialog box appears.

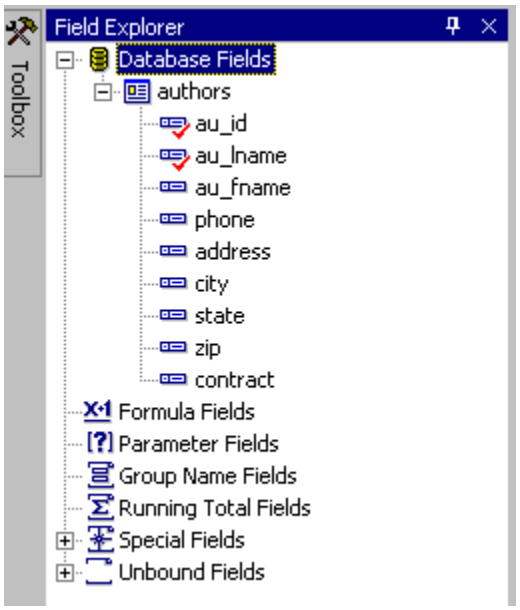


4. Drill down into **Project Data** > **ADO.NET DataSets** > **ADONET1DataSet1Table.Dataset1** > **authors**.

   Highlight **authors** and click the right arrow to move this table into the **Selected Tables** section. This will add the dataset to the report. Click **OK.**

5. In the **Field Explorer** expand **Database Fields** and select the desired fields from the **authors** table.

   Click and drag the desired fields from the **Field Explorer** to the details section of the report.



Preview the report

1. In the **Solution Explorer,** double-click **Form1.vb** to view the form.

   In the **Toolbox,** double-click **CrystalReportViewer** under the **Windows Forms** component to add **CrystalReportViewer1** to the form.

2. Click **CrystalReportViewer1** and press **F4** to go its Properties Window. Set the **Dock** property to **Fill**.

3. Highlight **Form1.vb** and click the **View Code** icon in the **Solution Explorer** to enter the code window.

   Add the following lines of code above the line, **Public Class Form1**:

```
"Engine library used for basic calls to the Crystal Reports engine
Imports CrystalDecisions.CrystalReports.Engine

"Shared library used for database logon and passing datasets
Imports CrystalDecisions.Shared

"These libraries used for database connection and dataset creation
Imports System.Data
Imports System.Data.OleDb
```

4. Add the following lines of code just below the line **Inherits System.Windows.Forms.Form**:

```
" crReportDocument object points to the report
Dim crReportDocument As New CrystalReport1()

" Use Dataset object to pass data into report using dataset
Dim DataSet1 As DataSet

" Use adoOleDbConnection to connect to the SQL database
Dim adoOleDbConnection As OleDbConnection

" Use adoOleDbDataAdapter to fill Dataset1 with data
Dim adoOleDbDataAdapter As OleDbDataAdapter
```

5. Add the following lines of code just below the line, **InitializeComponent():**

```
Dim connectionString As String = ""

"Enter the log on information for your database
connectionString = "Provider=SQLOLEDB;"
connectionString += "Server=DBCONN1;Database=pubs;"
connectionString += "User ID=userid;Password=password"

"Create and open a connection using the connection string
adoOleDbConnection = New OleDbConnection(connectionString)

"Build a SQL statement to query the datasource
Dim sqlString As String = ""
sqlString = "SELECT * FROM authors"

"Retrieve the data using the SQL statement
adoOleDbDataAdapter = New OleDbDataAdapter(sqlString,
adoOleDbConnection)

"Create a instance of a Dataset
DataSet1 = New DataSet()

"Fill the dataset with the data with author information.
"The table name used in the Fill method must be identical to the name
"of the table in the report.

adoOleDbDataAdapter.Fill(DataSet1, "authors")

"Pass the dataset to the report
crReportDocument.Database.Tables(0).SetDataSource(DataSet1)

"View the report
CrystalReportViewer1.ReportSource = crReportDocument
```

NOTE	It is very important that the structure of the SQL query (used to pass the data to the dataset) is identical to the structure of the dataset the report was created off of. If the SQL query and dataset structure are different, you may get a “Query Engine Error” when the report is run.

6. Run the project by clicking the **Start** button at the top toolbar of Visual Studio.

   A report with data now appears in the preview window.

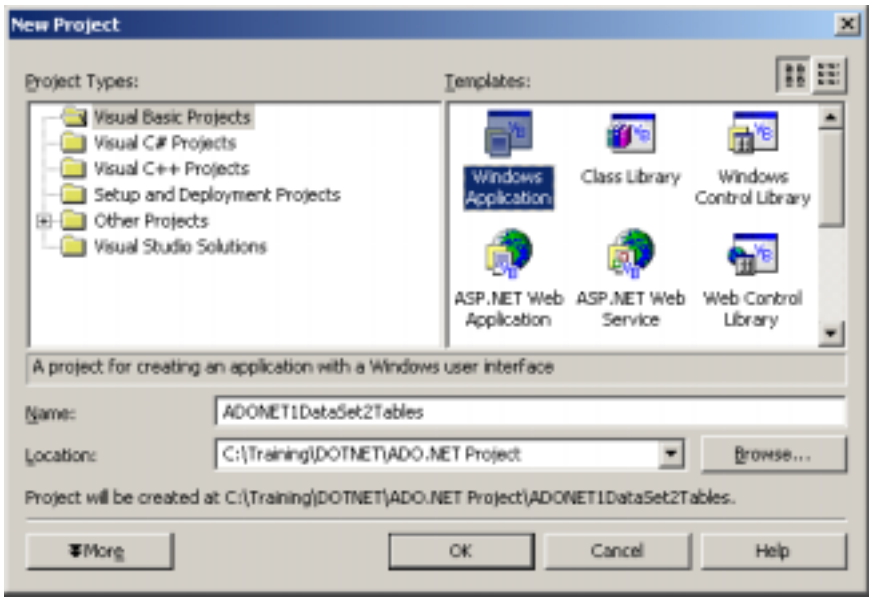
Tutorial 2 – Using a single DataSet and Multiple Tables

This tutorial explains how to create a report using Crystal Reports for Visual Studio .NET and ADO.NET. This tutorial describes creating a new report based on multiple tables from a single dataset file and passing the dataset to the report.

1. Start Visual Studio .NET and Select **New Project** from the Visual Studio .NET start page. The **New Project** dialog box appears.

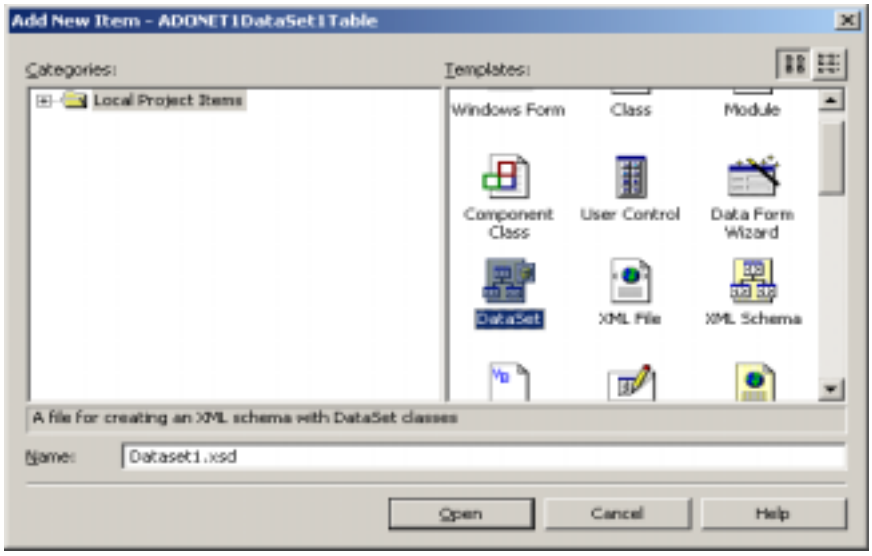
2. Select **Visual Basic Projects** as the Project Type and select **Windows Application** as the Template

Name the application, **ADONET1DataSet2Table**. Select **OK**. You should now see a VB form.



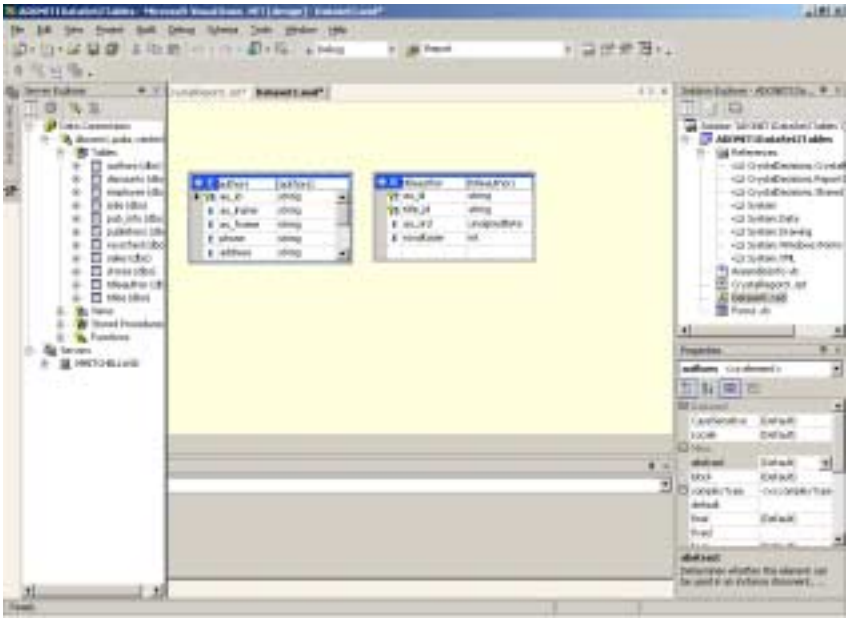
Add a dataset

1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **dataset** object and keep the default name, **Dataset1.xsd.** Select **Open**.



2. Once the dataset has been added to the project, browse through the **Server Explorer** to find tables to add to this dataset. In this example, find the Microsoft SQL Server sample database, **Pubs** and locate the **authors** and **titleauthor** tables.

   Drag the **authors and titleauthor** tables from the **Server Explorer** to the **dataset** tab of Dataset1.xsd.



3. On the **Schema** menu, click **Generate Dataset** to generate the dataset object for the project. In this scenario, the dataset object will contain the data description for the **authors** and **titleauthor** tables.

   The generated dataset object contains only a description of the database.

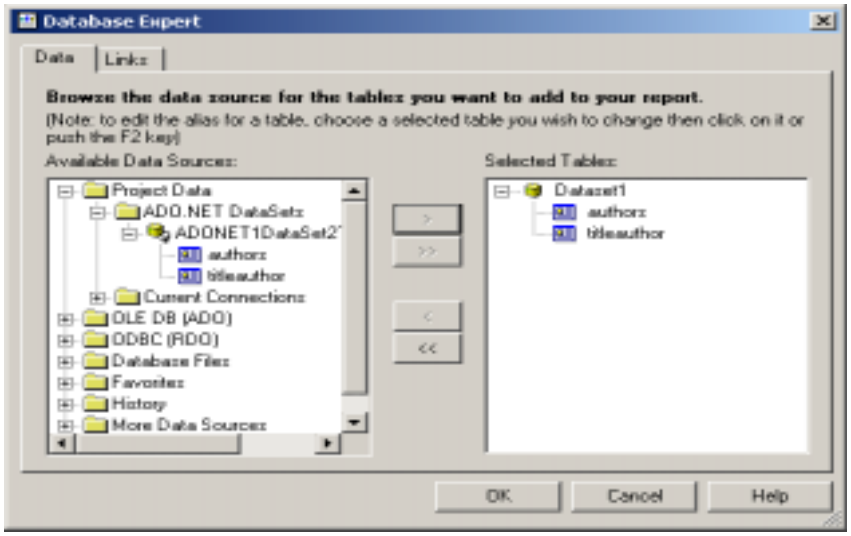
4. Save your dataset (dataset1.xsd).

Build a Report

1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **Crystal Reports** template and keep the default name, **CrystalReports1.rpt**. Click **Open**.

2. The **Crystal Report Gallery** appears. Select the **As Blank Report** option and click **OK**. The design view of the report appears including the **Field Explorer** on the left hand side.

3. In the **Field Explorer**, right-click the **Database Fields** option and select **Database Expert.** The **Database Expert** dialog box appears.



4. Drill down into **Project Data** > **ADO.NET DataSets** > **ADONET1DataSet2Table.dataset1** > **authors**.

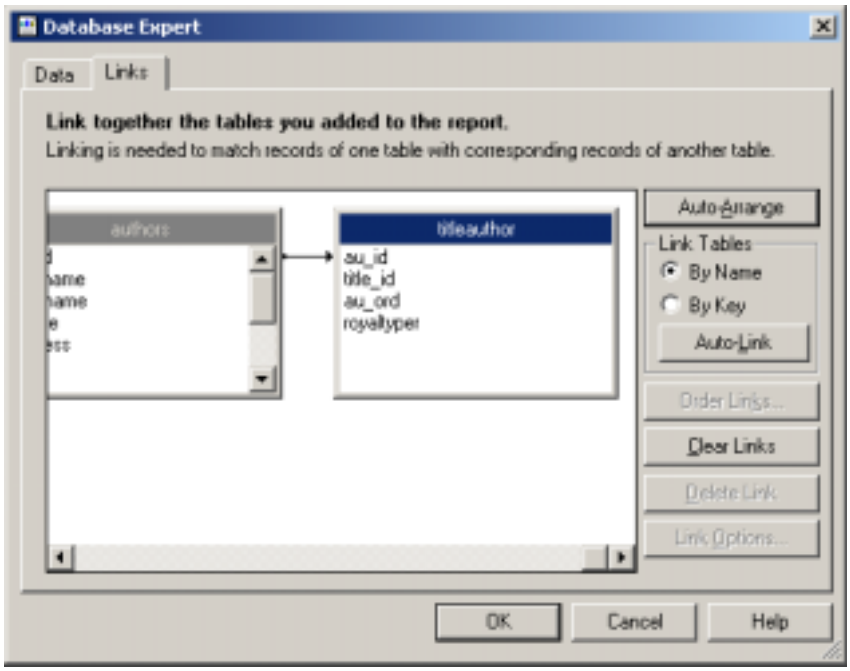
   Highlight **authors** and click the arrow to move this into the **Selected Tables** section.

   Move the **titleauthor** table into the **Selected Tables** section as well.

   Click **OK.** This will add the dataset to the report.

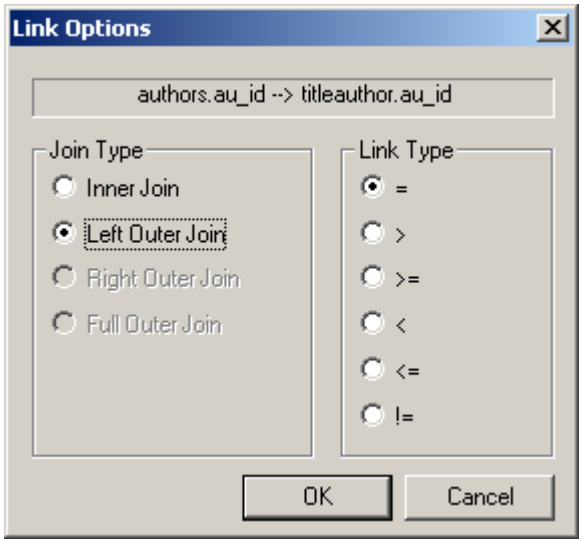
5. The **Database Expert** dialog box appears to prompt you to link the **authors** and **titleauthor** tables.

   The report will automatically create a link between these two tables. This link will be an inner join. Since there is a one to many relationship, change the link to be a left outer join.



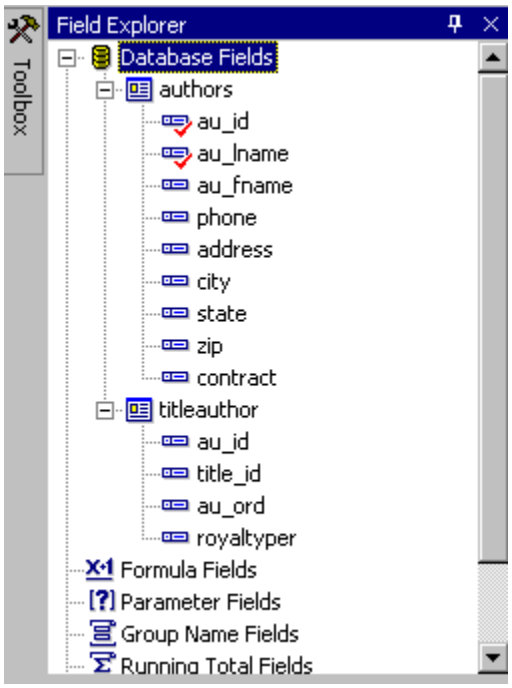
6. To change the link to a left out join, right-click on the arrow between the two tables and select **Link Options…** from the popup menu.

   In the **Link Options** dialog box, change the link from an **Inner Join** to a **Left Outer Join** and click **OK**.



7. In the **Field Explorer** expand **Database Fields** and select the desired fields from the **authors** and **titleauthor** tables.

   Click and drag the desired fields from the **Field Explorer** to the details section of the report.



Preview the Report

1. In the **Solution Explorer,** double-click **Form1.vb** to view the form.

   In the **Toolbox,** double-click **CrystalReportViewer** under the **Windows Forms** component to add **CrystalReportViewer1** to the form.

2. Click CrystalReportViewer1 and press **F4** to go its Properties Window. Set the **Dock** property to **Fill**.

3. Highlight **Form1.vb** and click the **View Code** icon in the **Solution Explorer** to enter the code window.

   Add the following lines of code above the line, **Public Class Form1**:

```
"Engine library used for basic calls to the Crystal Reports engine
Imports CrystalDecisions.CrystalReports.Engine

"Shared library used for database logon and passing datasets
Imports CrystalDecisions.Shared

"These libraries used for database connection and dataset creation
Imports System.Data
Imports System.Data.OleDb
```

4. Add the following lines of code just below the line **Inherits System.Windows.Forms.Form**:

```
" crReportDocument object points to the report
Dim crReportDocument As New CrystalReport1()
```

```
" Use Dataset object to pass data into report using dataset
Dim DataSet1 As DataSet

" Use adoOleDbConnection to connect to the SQL database
Dim adoOleDbConnection As OleDbConnection

" Use the 2 DataAdapter objects, one for each table that is passed to
" the report
Dim adoOleDbDataAdapter As OleDbDataAdapter
Dim adoOleDbDataAdapter2 As OleDbDataAdapter
```

5. Add the following lines of code just below the line, **InitializeComponent():**

```
Dim connectionString As String = ""

" Enter the log on information for your database
connectionString = "Provider=SQLOLEDB;"
connectionString += "Server=DBCONN1;Database=pubs;"
connectionString += "User ID=userid;Password=password"

"Create and open a connection using the connection string
adoOleDbConnection = New OleDbConnection(connectionString)

"Build a SQL statement to query for the authors table
Dim sqlString As String = ""
sqlString = "SELECT * FROM authors"

"Retrieve the data using the SQL statement
adoOleDbDataAdapter = New OleDbDataAdapter(sqlString,
adoOleDbConnection)

"Build a SQL statement to query for the titleauthor table
sqlString = "SELECT * FROM titleauthor"

"Retrieve the data using the SQL statement
adoOleDbDataAdapter2 = New OleDbDataAdapter(sqlString,
adoOleDbConnection)

"Create a instance of a Dataset
DataSet1 = New DataSet()

"Fill the dataset with the data with author information
adoOleDbDataAdapter.Fill(DataSet1, "authors")

"Fill the dataset with the data with titleauthor information.
"The table name used in the Fill method must be identical to the name
"of the table in the report.

adoOleDbDataAdapter2.Fill(DataSet1, "titleauthor")

"Pass the dataset to the report
crReportDocument.Database.Tables(0).SetDataSource(DataSet1)

"View the report
CrystalReportViewer1.ReportSource = crReportDocument
```

NOTE	It is very important that the structure of the SQL query used to pass the data to the dataset is identical to the structure of the dataset the report was created off of. If the SQL query and dataset structure are different, you may get a ““Query Engine Error”.

6. Run the project by clicking the **Start** button at the top toolbar of Visual Studio.

   A report with data now appears in the preview window.

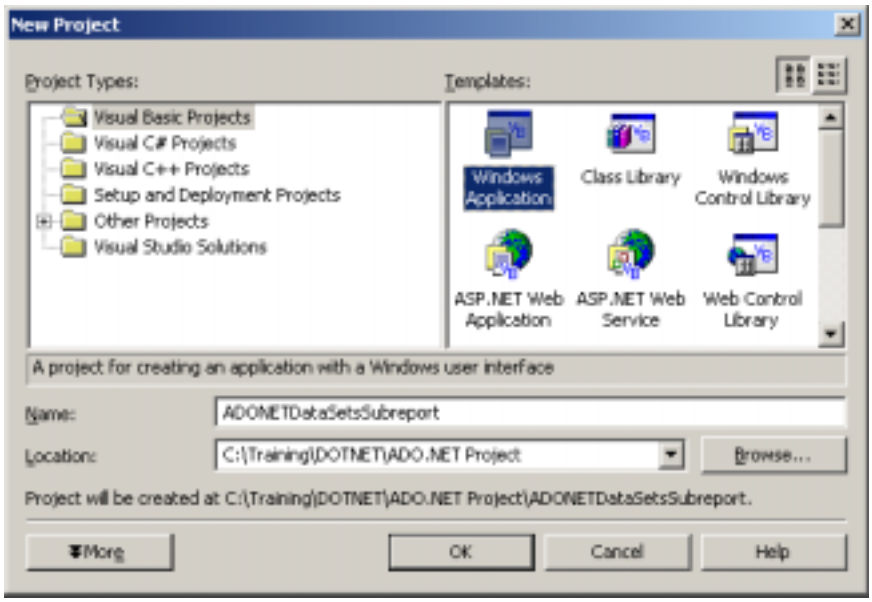
Tutorial 3 – Using multiple Datasets with a Main and Subreport

This tutorial explains how to create a report using Crystal Reports for Visual Studio .NET and ADO.NET. This tutorial describes creating a main report and a subreport based on separate datasets.

1. Start Visual Studio .NET and Select **New Project** from the Visual Studio .NET start page. The **New Project** dialog box appears.

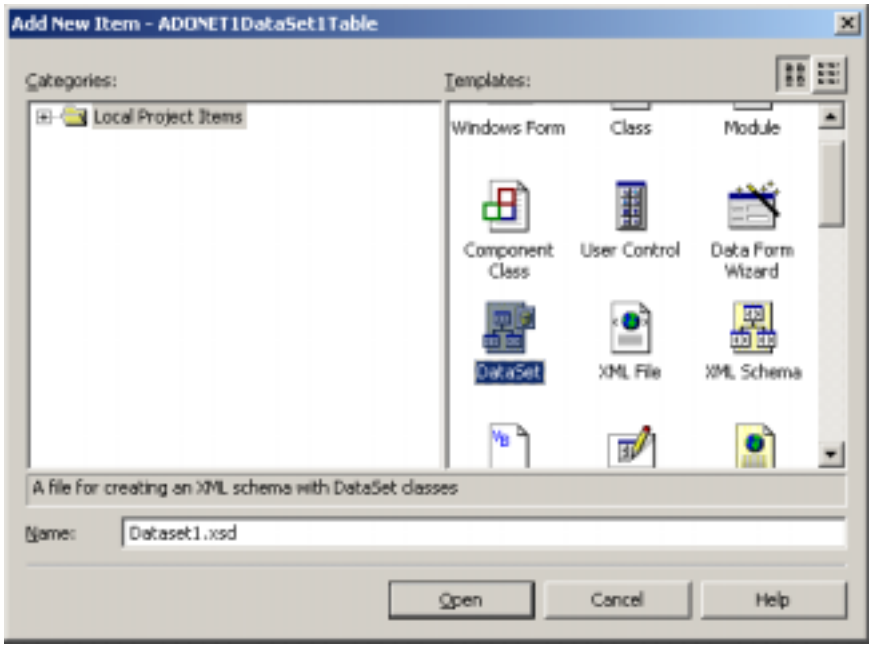
2. Select **Visual Basic Projects** as the Project Type and select **Windows Application** as the Template.

Name the application, **ADONETDataSetsSubreport**. Select **OK**. You should now see a VB form.



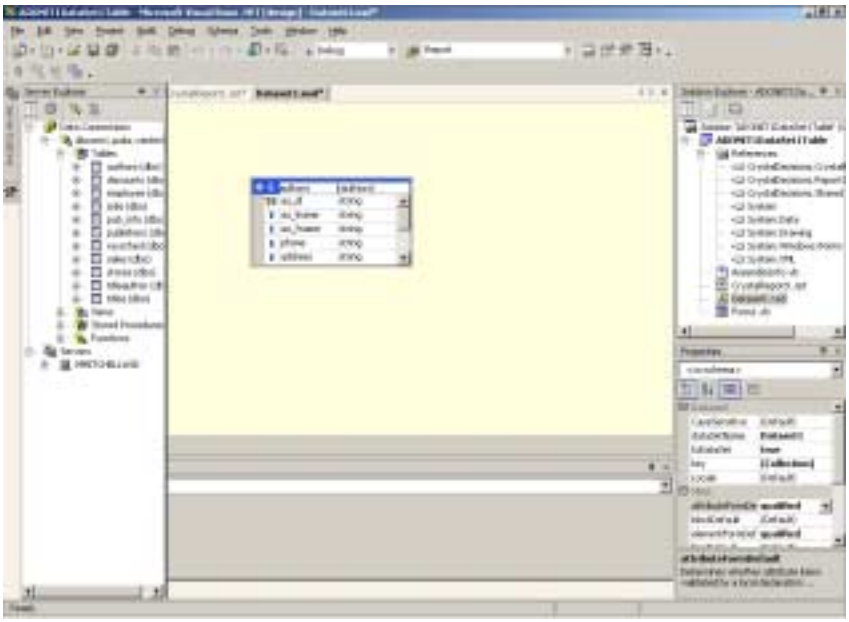
Add a dataset for the main report

1. Go to the **Project** menu and select **Add New Item.…**. The **Add New Item** dialog box appears. Select a **dataset** object and keep the default name, **Dataset1.xsd.** Select **Open**.



2. Once the dataset has been added to the project, browse through the **Server Explorer** to find tables to add to this dataset. In this example, find the Microsoft SQL Server sample database, **Pubs** and locate the **authors** table**.**

   Drag the **authors** tables from the **Server Explorer** to the **dataset** tab of Dataset1.xsd.



3. On the **Schema** menu, click **Generate Dataset** to generate the dataset object for the project. In this scenario, the dataset object will contain the data description for the **authors** table.

   The generated dataset object contains only a description of the database.

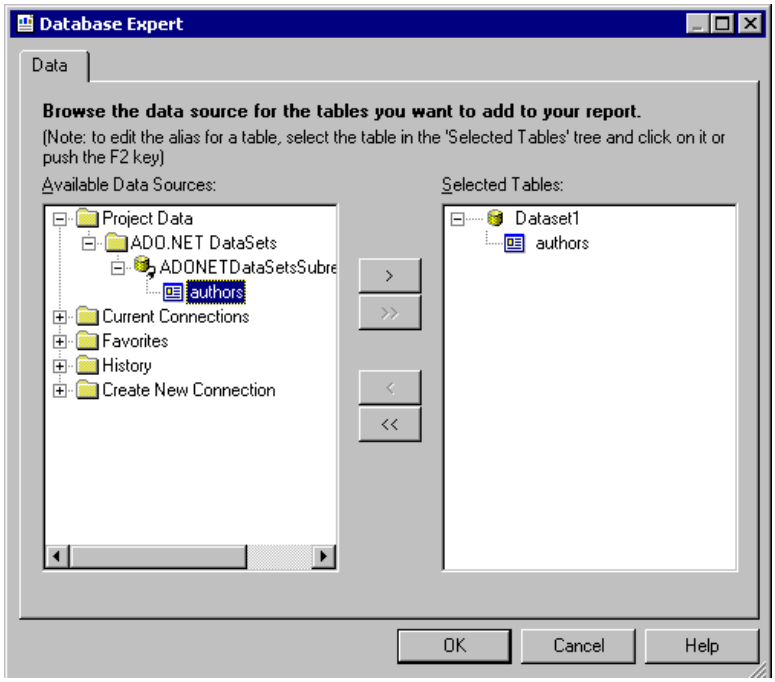
4. Save your dataset (dataset1.xsd).

Build the main report

1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **Crystal Reports** template and keep the default name, **CrystalReports1.rpt**. Click **Open**.

2. The **Crystal Report Gallery** appears. Select the **As Blank Report** option and click **OK**. The design view of the report appears including the **Field Explorer** on the left hand side.

3. In **the Field Explorer**, right-click the **Database Fields** option and select **Database Expert.** The **Database Expert** dialog box appears



6. Drill down into **Project Data** > **ADO.NET DataSets** > **ADONETDataSetsSubreport.Dataset1** > **authors**.

   Highlight **authors** and click the right arrow to move this into the **Selected Tables** section. This will add the dataset to the report. Click **OK.**

7. In the **Field Explorer** expand **Database Fields** and select the desired fields from the **authors** table.

   Click and drag the desired fields from the **Field Explorer** to the details section of the report.

Add a second dataset for the subreport

1. Go to the **Project** menu and select **Add New Item…**. The **Add New Item** dialog box appears. Select a **dataset** object and keep the default name, **Dataset2.xsd.** Select **Open**.

2. Once the dataset has been added to the project, browse through the **Server Explorer** to find a table to add to this dataset. In this example, find the Microsoft SQL Server sample database, **Pubs** and locate the **titleauthor** table**.**

   Drag the **titleauthor** tables from the **Server Explorer** to the **dataset** tab of Dataset2.xsd.

3. On the **Schema** menu, click **Generate Dataset** to generate the dataset object for the project. In this scenario, the dataset object will contain the data description for the **titleauthor** table.

   The generated dataset object contains only a description of the database.

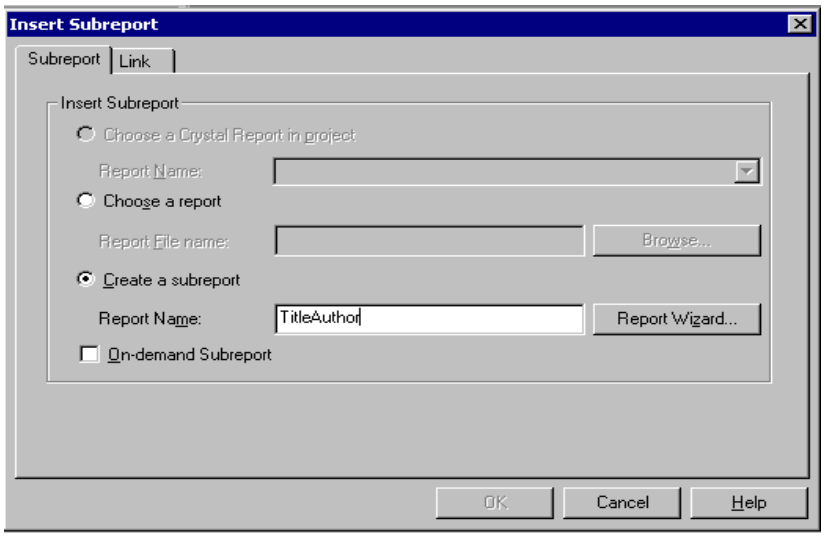
4. Save your dataset (dataset2.xsd).

Add a subreport

1. Right-click in the white space of the detail section of the main report, and from the pop-up menu, select **Insert** then **Subreport**.

   Use your mouse to indicate where you want to place the subreport.

2. The **Insert Subreport** dialog box appears. Name the subreport, **TitleAuthor** and click the **Report Wizard** button.



3. The **Standard Report Creation Wizard** appears. Under **Available Data Sources,** drill down into **Project Data** > **ADO.NET DataSets** > **ADONETDataSetsSubreport.Dataset2**> **titleauthor**.

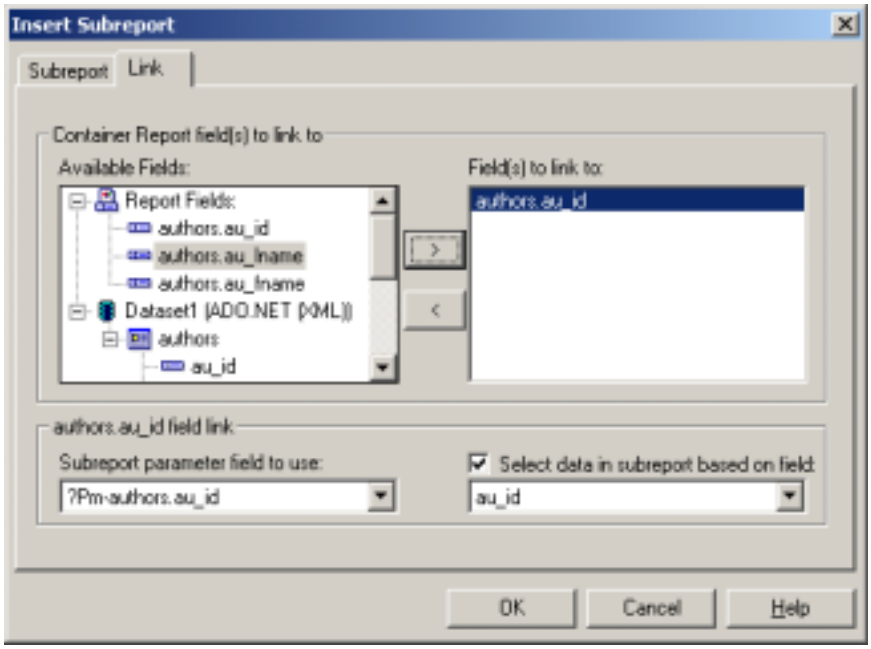
4. Select **titleauthor** and use the right arrow to insert **titleauthor** as a **Selected Table.** Click **Next.**

5. Add the fields you would like to display on the subreport and then click **Finish.**

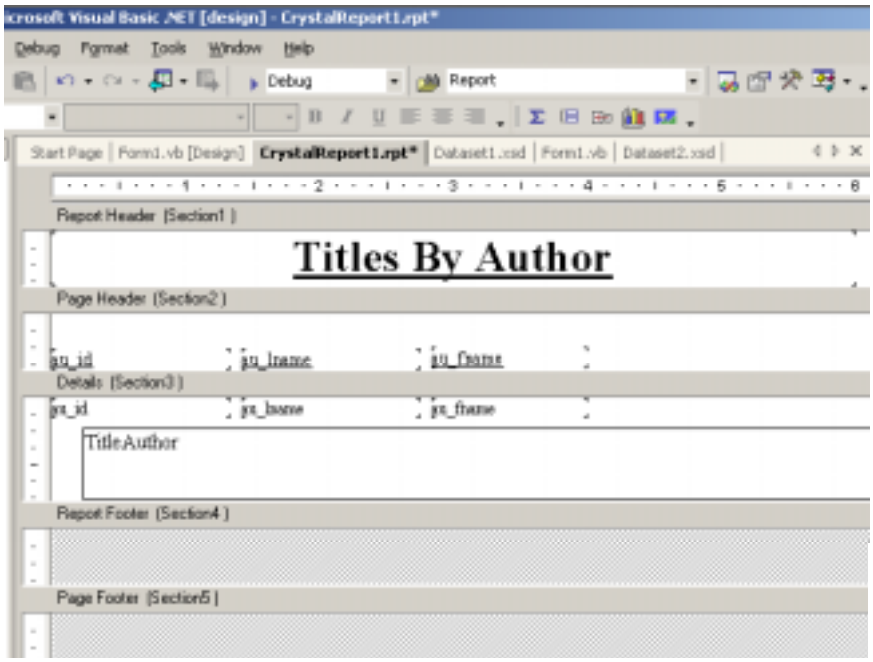
6. At the **Insert Subreport** dialog box, click the **Link** tab. Highlight **authors.au_id** and click the right arrow to move it to the **Field(s) to link to** section.

   This will create a link between the main report and the subreport based on the **au_id** field.

7. Click **OK** to go back to the report.



You should now have a report that looks similar to the following:



Preview the Report

1. In the **Solution Explorer,** double-click **Form1.vb** to view the form.

   In the **Toolbox,** double-click **CrystalReportViewer** under the **Windows Forms** component to add **CrystalReportViewer1** to the form.

2. Click **CrystalReportViewer1** and press **F4** to go its Properties Window. Set the **Dock** property to **Fill**.

3. Highlight **Form1.vb** and click the **View Code** icon in the **Solution Explorer** to enter the code window.

   Add the following lines of code above the line, **Public Class Form1**:

```
"Engine library used for basic calls to the Crystal Reports engine
Imports CrystalDecisions.CrystalReports.Engine

"Shared library used for database logon and passing datasets
Imports CrystalDecisions.Shared

"These libraries used for database connection and dataset creation
Imports System.Data
Imports System.Data.OleDb
```

4. Add the following lines of code just below the line **Inherits System.Windows.Forms.Form**:

```
" The Document objects points to the main and subreport
Dim crReportDocument As New CrystalReport1()
Dim crSubReportDocument As New ReportDocument()

" Use Dataset objects to pass data into the reports
Dim DataSet1 As DataSet
Dim DataSet2 As DataSet

" Use adoOleDbConnection to connect to the SQL database
Dim adoOleDbConnection As OleDbConnection

" Use the 2 DataAdapter objects, one for each table that is passed to
" the report
Dim adoOleDbDataAdapter As OleDbDataAdapter
Dim adoOleDbDataAdapter2 As OleDbDataAdapter
```

5. Add the following lines of code just below the line, **InitializeComponent():**

```
Dim connectionString As String = ""

" Enter the log on information for your database
connectionString = "Provider=SQLOLEDB;"
connectionString += "Server=DBCONN1;Database=pubs;"
connectionString += "User ID=userid;Password=password"

"Create and open a connection using the connection string
adoOleDbConnection = New OleDbConnection(connectionString)

"Build a SQL statement to query the datasource
Dim sqlString As String = ""
sqlString = "SELECT * FROM authors"
```

```
"Retrieve the data using the SQL statement
adoOleDbDataAdapter = New OleDbDataAdapter(sqlString,
adoOleDbConnection)

"Create an instance of a Dataset
DataSet1 = New DataSet()

"Fill the dataset with the data retrieved
adoOleDbDataAdapter.Fill(DataSet1, "authors")

"Create an instance of a Dataset for the subreport
DataSet2 = New DataSet()

"Retrieve the data using the SQL statement and existing connection
sqlString = "SELECT * FROM titleauthor"
adoOleDbDataAdapter2 = New OleDbDataAdapter(sqlString,
adoOleDbConnection)

"Fill the dataset with data for the subreport
adoOleDbDataAdapter2.Fill(DataSet2, "titleauthor")

"Open the subreport so that we can pass a dataset to it.
crSubReportDocument =
crReportDocument.OpenSubreport("TitleAuthor")

"Pass the datasets to both the main report and subreport objects
crReportDocument.Database.Tables(0).SetDataSource(DataSet1)
crSubReportDocument.Database.Tables(0).SetDataSource(DataSet2)

" view the report
CrystalReportViewer1.ReportSource = crReportDocument
```

NOTE	It is very important that the structure of the SQL query (used to pass the data to the dataset) is identical to the structure of the dataset the report was created off of. If the SQL query and dataset structure are different, you may get a “Query Engine Error” when the report is run.

6. Run the project by clicking the **Start** button at the top toolbar of Visual Studio.

   A report with data now appears in the preview window.

Additional Resources

- For another tutorial for reporting off and ADO.NET dataset, go to http://support.crystaldecisions.com/docs and download:

  rtm_reportingoffadonetdatasets.pdf

- For sample applications demonstrating using ADO and ADO.NET, go to http://support.crystaldecisions.com/downloads and download:

  - **vbnet_win_adodotnet.exe** (populate an ADO.NET dataset and pass the dataset to a report at runtime)

  .

- **vbnet_win_classicado.exe** (populate a classic ADO recordset and pass the recordset to a report at runtime)

- For knowledge base articles related to ADO.NET, go to http://support.crystaldecisions.com/kbase and search for the keyword **'ado.net'.**

Contacting Crystal Decisions for Technical Support

Along with this document, we recommend that you visit our Technical Support web site for further resources and sample files. For further assistance, visit us at the web sites 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