

# RBDMS Report Creation

<b>The Work Area.....</b>	<b>2</b>
<b>Create the Project .....</b>	<b>3</b>
<b>Add a Report to the Project .....</b>	<b>4</b>
<b>Set the Report Data Sources.....</b>	<b>4</b>
<b>Add Header and Footer Information.....</b>	<b>4</b>
<b>Develop the Report Body .....</b>	<b>5</b>
<b>Add Groups for Sorting.....</b>	<b>5</b>
<b>Add Eye Candy.....</b>	<b>7</b>
Background color .....	7
Images.....	7
<b>Create Report Parameters .....</b>	<b>8</b>
<b>Add Report to the Application Menu, Create and Associate Filters.....</b>	<b>9</b>
Create a Filter Set for the Report.....	10
Notes for Creating Filter Controls .....	10
Create Menu Item .....	11
<b>Test and Debug the Report .....</b>	<b>12</b>
<b>About Subreports.....</b>	<b>12</b>
<b>Recommended Reading.....</b>	<b>12</b>

The following practice exercise contains a series of tips and hints for creating .rdlc-formatted reports and then adding them to RBDMS applications through RBDMS WinAdmin.NET. Although this document is not meant to replace the voluminous Microsoft Visual Studio help system, nor the many fine commercially produced tomes on the market, it will provide new users some getting-started advice and familiarize them with some conventions the developers have adopted for RBDMS.NET applications.

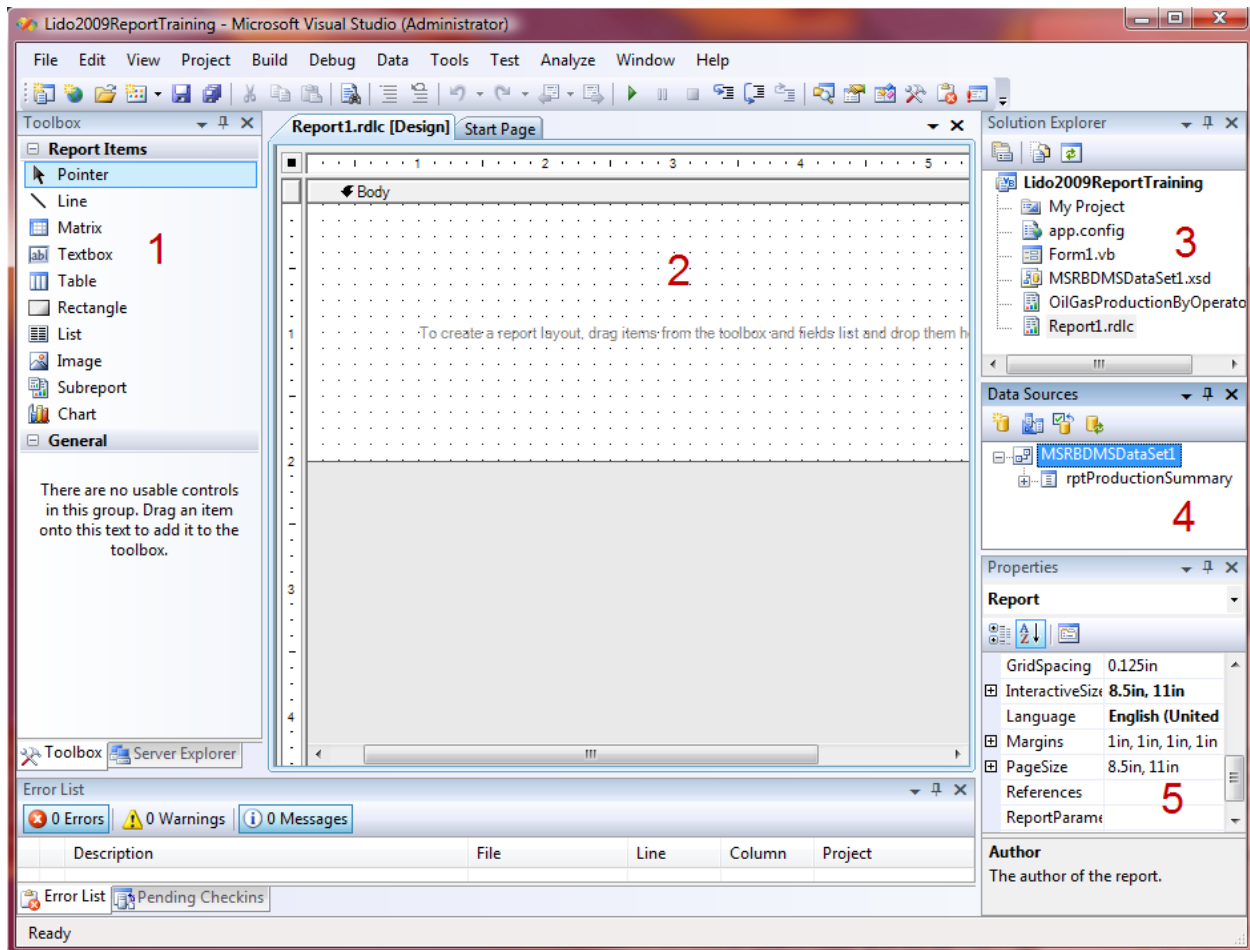
Reports are usually defined through three components:

- A query (SELECT, WHERE, ORDER BY) run against a SQL Server database or a data view within SQL Server. Multiple table adapters that describe a set of these queries or views can be defined as a dataset within a Reports project in VisualStudio.NET 2008. This dataset specifies the data source for purposes of report data binding.
- A filter that contains elements compatible with the query or view and that is associated with the report.
- A template with data fields bound to the fields exposed in the table adapter or view.

## The Work Area

Within Visual Studio 2008, the following components of the workspace are of importance:

1. The toolbox of standard report controls.
2. The work space, which allows drag-and-drop, page layout, and spreadsheet-like action for tables, charts, and letter reports.
3. The Solution Explorer, which is used to manage project content.
4. The Data Sources pane, which is used to manage the database objects referenced in the report project.
5. The Properties pane, which is used to manage formatting requirements at both the page and the individual control level.



In the following exercise, you will create a report project in Visual Studio and develop a report with the database objects in your local installation of the MSRBDMS.NET database. Then you will add the report to your local copy of the front-end application, building the necessary filter set and menu objects within RBDMS WinAdmin.NET.

Visual Studio includes an option to create a ReportApplication with the MainForm and Report1.rdlc items created for you. When you create a ReportApplication, many of the steps to start a report template are accomplished via a wizard. However, in this exercise, you will create a WinForm application and step through the process of creating reports from A to Z. Although some people find the wizard easier to use, some options you may otherwise exercise are taken away, and since the purpose is to further an understanding of the overall process, this document takes you through the steps.

**What's the difference between .rdl and .rdlc files?** Report language definition (.rdl) files are created with the SQL Server 2005 Report Designer and are meant to be run from the Report Server. Report language definition client (.rdlc) files are created with VS2008 and are client side. RDLC files require the ReportViewer control, and no preview is available without running the project. However, .rdl and .rdlc formats may be converted back and forth, and the RBDMS Reports module can accommodate either format. Projects must be WinForm to use the VS2008 ReportViewer. WPF can use the ReportViewer control, but only in Windows compatibility mode.

The .rdlc format is preferred over the .rdl format typically used with SQL Server Reporting Services installations because some agencies prefer not to run SQL Server Reporting Services, either because they are still running SQL Server 2000 or because running SQL Server Reporting Services is problematic in some other way.

This document covers only the client-side processing of reports, since the .rdlc format has proven to be easier to use with RBDMS applications. More options are available with SQL Server Reporting Server, but are beyond the scope of this document. Likewise, this document does not include the steps to run or preview .rdlc reports from within Visual Studio, which requires a separate window with a ReportViewer control to host the .rdlc report.

## **Create the Project**

1. In Visual Studio, click **File | New Project** and create a new Windows Form Application project and name it **Lido 2009ReportTraining**.
2. Click **File | Save All**. Set the path to the solution folder and click **Save**.
3. From the **Data** menu, select **Add New Data Source**.
4. When prompted to choose a data source type, click **Database** and **Next**.
5. Click **New Connection** and configure the datasource for your local MSRBDMS connection with Windows Authentication. Test the connection and click **OK**. Click **Next**.
6. Save the connection string to the app config file. Click **Next**.
7. Choose your database objects. From **Views**, choose *rptProductionSummary* to create a dataset for the project.

## ***Add a Report to the Project***

1. In the Solution Explorer, right click the project, select **Add | New Item**.
2. In the **Add New Item** dialog, choose **Report** and name the new .rdlc *OilGasProductionByOperator*.

## ***Set the Report Data Sources***

1. From the **Report** menu, select **Data Sources**.
2. From the **Project Data Sources** in the **ReportData Sources** dialog, select the object to which you would like to bind the report (rptProductionSummary). Click **Add to Report**. Click **OK**.

**Hint:** Sometimes while working with report projects, you will find that you need to edit the data underlying a particular report to add fields or to streamline a query. If you must update a table adapter that you are using as the data source for binding data to a report (datasetname.xsd in the report project) or if you change an underlying view that the report is based on in SQL Server (e.g., if you wish to include additional fields), follow these steps:

1. Make the desired change in either the dataset in Visual Studio or the view in SQL Server, as appropriate, and save your changes.
2. In Visual Studio, open the report you want to update with the new view.
3. Refresh the data source for the report within Visual Studio.
  - a. From the menu, select **Reports | Data Sources**. The **Reports Data Sources** dialog will open.
  - b. Select the data source you want to refresh and click **Refresh All**.

**Note:** If the project does not contain any Server Connections or no datasets are defined, Visual Studio will launch a Dataset Configuration Wizard, which will go through the steps necessary to connect to the SQL Server data source. Other datasources besides SQL Server are available, but only SQL Server connections should be used in RBDMS.

## ***Add Header and Footer Information***

1. From the **Report** menu, select **Page Header** to view the page header space.
2. Drag a textbox from the Toolbox into the header space and enter a title for the report "Production by Operator."
3. Set the formatting in the **Properties** window for the textbox. Set for bold, 14-point font.

4. Drag four more text boxes to set *From:* and *To:* dates for the report run. We'll set the report parameters later.
5. Add footer information: From the **Report** menu, select **Page Footer** to view the page footer space.
6. Drag two textboxes from the Toolbox into the footer space and enter `=Now()` for the report run date in one and `= "Page " & Globals.PageNumber & " of " & Globals.TotalPages` to print page numbers.

**Hint:** Headers and footers may be set to display on every page of the report, or be excluded from the first and/or last page. This option is a property of the Header/Footer which can be set in the Properties pane. Headers and footers should be used only when the report is long enough to span multiple pages.

## Develop the Report Body

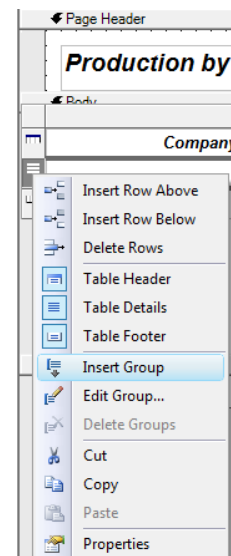
1. Drag a table control onto the main field.
2. Format the grid to include five columns with the headings *Company*, *Well ID*, *Report Period*, *Oil (bbls)*, and *Gas (mcf)*.
3. Add desired formatting to these column headers and text alignment through the **Properties** pane.






**Hint:** The first control you drag onto the Visual Studio work surface should **always** be a List box, a Table, or a Matrix control. Build the body of the report inside the list box or within the table. List boxes and tables allow the report to “grow,” so if you fail build the report with one of these controls, the report will stop printing after the first record. While it is not impossible to add a list box after the template is in a state of advanced development, it is awkward and, in short, no fun.

## Add Groups for Sorting

Many reports include tabular totals that require specifying a grouping and sort order. For these reports, you must be sure to specify the sort/group on the **Grouping and Sorting Properties** dialog, which is available only as a right-click from the line in the table you are grouping on within the Visual Studio workspace.

1. Select the second table row, right click, and insert a new group.



2. In the **Grouping and Sorting** dialog box on the **General** tab, **Name** field, enter *Company*.
3. In the **Group on:** grid, select the *Operator.Value* from the dropdown.
4. Switch to the **Sorting** tab. Under **Expression**, select *Operator.Value*. The **Direction** will default to *Ascending*. Change it if you wish.
5. Click **OK**.
6. Select the cell under **Company** column header In the row marked with the group 1 icon  and in the textbox **Properties, Value** field, select from the dropdown list *=Fields!OperatorName.Value*.
7. Repeat the steps in 1-6 above to add a second group for grouping and sorting by *WellID.Value*.
8. In the details row (delineated with the row header icon ) , under the Report Period column header, bind the field to the ReportPeriod by typing in the cell or selecting *=Fields!ReportPeriod.Value* from the Value field in the Properties pane formatted for a short date (**d**) display. Also on the details line, bind the columns for Oil and Gas to *=Fields!OilProd.Value* and *=Fields!GasProdValue*, respectively. Specify the format for these fields as **n0** in the Properties pane.
9. Add a totals row for group 2: On the row marked with the  row header, enter the words *Subtotals:* and then, for each product column, enter code to summarize the data in the following format: *=Sum(Fields!<fieldname>.Value)*. Specify a data format of **n0** in the Properties pane.
10. Add a totals row for group 1: On the row marked with the  row header, enter the words *Company Totals:* and then, for each product column, enter code to summarize the data in the following format: *=Sum(Fields!<fieldname>.Value)* in the Properties pane. Format each of these fields as **n0**.
11. On the last row of the report, which is marked by the row header icon , enter the words **Report Totals:** and then, for each product column, repeat the summation statement and format specification above.

## Add Eye Candy

### Background color

For long grid reports, you may wish to introduce some eye-relief. To alternate background color on the report grid, use an “iif” expression for the BackgroundColor property for the row where you want the alternating color to begin, like so:

```
=iif(LineNumber(Nothing) Mod 2, "AliceBlue", "White")
```

### Images

Adding an image to an .rdlc is a two-step process: embedding the asset and then calling it from the **Properties** pane. Adding an image via the Embedding option will copy the content of the selected image into the .rdlc report.

Although it is also possible to create a “Link” to an external image to be used by the report, additional programming is required and the EnableExternalImages property must be set to true on the RBDMS ReportViewer. Therefore, this method is not directly supported.

1. From the **Report** menu, choose **Embedded Images** to open the **Embedded Images** dialog box.
2. From the new record row, click **New Image** and navigate to a logo or other small image on your computer and select it. You can add more images at this point if you need to.
3. Then click **OK** to close the **Embedded Images** dialog.
4. Drag the **Image** control from the toolbox to the desired position on the workspace.
5. In the **Properties** pane, *Source* field, choose **Embedded** from the dropdown.
6. In the **Properties** pane, the *Value* field will be populated with a list of the images you embedded. Pick the one you want.

**Hints for controlling the layout:** If you find that you are having problems with the layout stretching horizontally off your targeted paper size for printing, be aware of two factors in your debugging:

1. Make sure you turn OFF the “allow to grow” option on the individual control properties on your report unless that is desirable behavior.
2. The reality is that an .rdlc template is basically a text file. Controlling the placement is often a matter of squish, jiggle, check, resquish, jiggle again, recheck, and cuss.

## Create Report Parameters

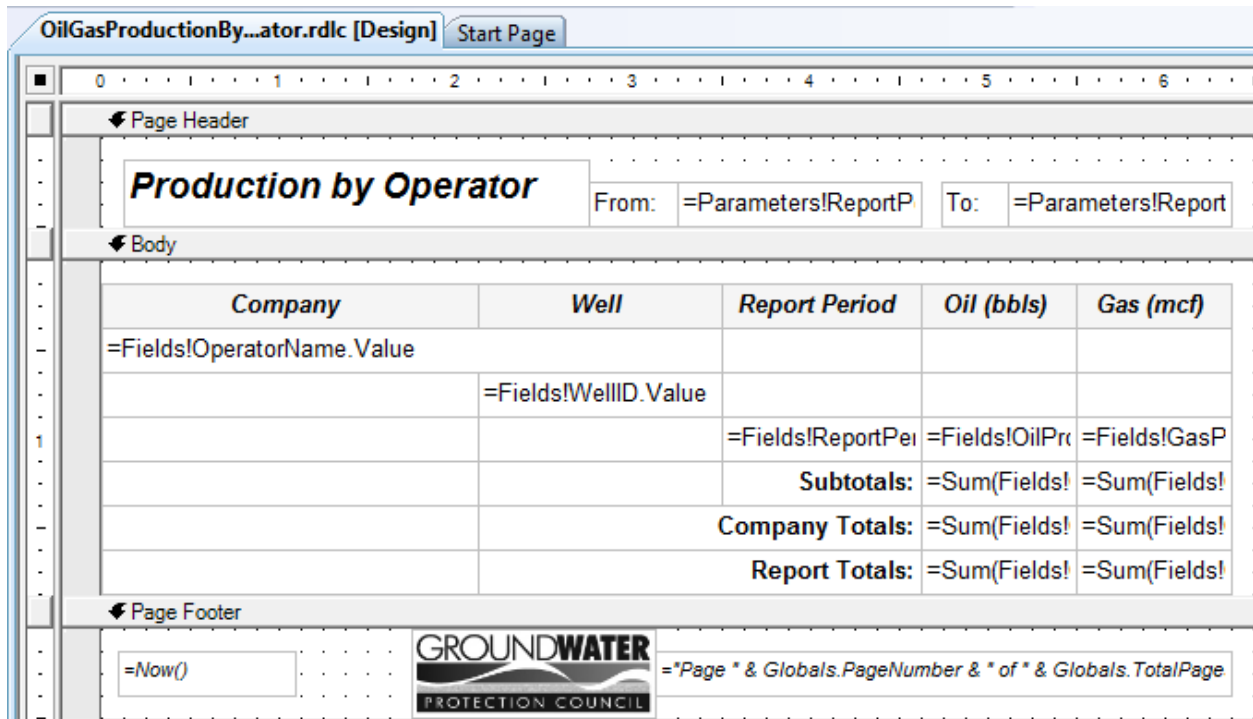
As a convention, the RBDMS.NET developers use the report parameter *CollName* to specify the name of the filter set created on the **Edit Filters** page in WinAdmin.NET. The report parameter *CollName* should be added to the dialog box accessed by the Visual Studio **Report | Report Parameters** menu call. The filter set name should be similar enough to the report name that it can be easily identified as being associated with the report when you are working with it from within WinAdmin.NET.

Any other report parameter that you define to be passed to the .rdlc (e.g., start and end dates, etc.) also must be defined as a filter control in WinAdmin.NET and assigned to the *CollName* (NameYouChose) filter set on the **Edit Filters** page of WinAdmin.NET.

To continue the exercise:

1. From the **Report** menu, select **Report Parameters** to open the **Report Parameters** dialog box.
2. Click **Add**.
3. In the **Properties Name** field enter *ReportPeriodStart* and change the data type to *DateTime*.
4. Click **Add**.
5. In the **Properties Name** field enter *ReportPeriodEnd* and change the data type to *DateTime*.
6. Click **Add**.
7. In the **Properties Name** field enter *CollName* as data type=*String*. Unclick the *Null* checkbox and enter *OperatorProduction* in the function field. This will be the name of the filter set you will associate with the report in WinAdmin.NET.
8. Return to the report header. In the empty text box next to **From:** enter *=Parameters!ReportPeriodStart.Value*. In the textbox next to **To:**, enter *=Parameters!ReportPeriodEnd.Value*. Specify a short date format (*d*) for these fields in the **Format** field in the **Properties** panel.
9. Save your work. Your report should look something like the following screen capture:





It is time to add the report to the application through RBDMS WinAdmin.NET for the smoke test.

## ***Add Report to the Application Menu, Create and Associate Filters***

The dynamic filtering parameters associated with each report are defined in RBDMS WinAdmin.NET on the **Edit Filters** form. These filter controls give you the ability to limit the results returned by the query underlying the report to the dataset of interest.

1. From the **Start** menu, select **All Programs | GWPC | RBDMS WinAdmin**.
2. In WinAdmin.NET, make sure that your database connection is pointed to your local installation of MSRBDMS on the **Forms | Configuration Options** page, **Database Connection** tab. If it is not already pointing to the local installation, reset the connections
3. With Windows Explorer, copy the .rdlc file you created into the folder shown for the appropriate report path, as shown in **Configuration | Settings** page.

## Create a Filter Set for the Report

1. In WinAdmin.NET, select **Forms | Edit Filters** to open the two-paneled page that is used to manage report filter controls.
2. In the left pane on the **New Record** row, enter the CollName you specified as a report parameter in Visual Studio (*OperatorProduction*). Enter a description for the filter collection in the *Description* column.
3. Build the filter collection to include *ReportPeriodStart* and *ReportPeriodEnd* along with any other filter control that your users will need to retrieve information by selecting filter controls on the right and then dragging and dropping them on the **Set Name** record for **OperatorProduction**. For this example, include the control **APINumber** and **OperatorNumber**. Right click the **Set Name** record to preview the filter.

**Hint:** You may delete items from your set on the left pane without deleting the control from the database. However, deleting filter controls from the right pane is a permanent action.

## Notes for Creating Filter Controls

If you must create a new filter control on the right pane of the **Edit Filters** page in RBDMS WinAdmin.NET, the following data dictionary snippet may provide useful hints in how to complete each new control record. The snippet shows the columns in the DevControls table, which governs the filtering in RBDMS.NET:

Column Name	Data Type	Length	Nulls	Default	Description
ControlName	varchar	50	NO		Unique name for the control (i.e., "Operator")
xml	text	2147483647	YES		Contains the XML serialization of the control object
ColumnName	varchar	50	YES		Column name for this control. Can include table name if needed.
Prompt	varchar	50	YES		The label to create next to the left of the comparison and criteria controls.
DataType	varchar	255	YES		The data type for the criteria ("String", "Integer", "DateTime", "Single", "Double")
ControlType	varchar	255	YES		The control type to use ("TextBox", "ComboBox", "CheckBox", "DateCombo", "ListBox")
DefaultValue	varchar	255	YES		A default value to use for the criteria.
Tooltip	varchar	255	YES		A tooltip to show when the cursor is over the control. Useful when you need a longer description than will fit in

Column Name	Data Type	Length	Nulls	Default	Description
					the label area.
CompareItems	varchar	255	YES		A semicolon delimited list of SQL comparison operators to make available (<; <=;;>=;;>;>;Like)
DefaultCompare	varchar	10	YES		The default comparison criteria (i.e., "=").
HideCompare	bit	1	YES	((0))	If true then the comparison control will be hidden. This is useful if the default value for the comparison control and you don't want the user to change it.
InputMask	varchar	50	YES		A data input mask to use for a textbox control.
MinValue	varchar	50	YES		The minimum allowable value.
MaxValue	varchar	50	YES		The maximum allowable value.
SQL	varchar	1024	YES		A SQL statement to retrieve data to display in a combo or listbox.
DisplayField	smallint	2	YES		The position of the description field (0 based).
ValueField	smallint	2	YES		The position of the value field (0 based)
ListItems	text	2147483647	YES		A semicolon delimited list of items to use in a combo or listbox.
Hidden	bit	1	YES		True if the control is hidden and always used with a default value and DefaultCompare
Format	varchar	50	YES		Format specification
Sort	bit	1	YES	((0))	If true then this field will be available for sorting.
Required	bit	1	YES	((0))	
ParameterOnly	bit	1	YES	((0))	If true this criteria will be passed as a parameter only and not part of the WHERE clause.

## Create Menu Item

1. In RBDMS WinAdmin.NET, switch to the **Security** tab of the **Menus and Security** page.
2. In the **Rights** pane, scroll to the **New Record** row and specify the right for the **OilGasProductionByOperator** report.
3. Expand the Rights record and assign the *Everyone* role to the right. Be sure to click off the new record to commit the write action.

4. Switch to the **Menus** tab of the **Menus and Security** page.
5. In the **Select Menu** dropdown, select *Reports*. This menu is the live menu for the RBDMS.NET WPF **Reports** menu as well as for the **Reports** page of RBDMS WinAdmin.NET.
6. Add the **OilGasProductionByOperator.rdlc** to the menu by selecting the record in the right pane and dragging it to the Well Data node in the left pane.
7. Click **Save Updates**.

## ***Test and Debug the Report***

1. From the **Forms** menu, select *Reports*.
2. Expand the **Well Data** node to see whether your new report is visible.
3. Either run or debug the report.

## ***About Subreports***

Subreports must be developed as separate .rdlcs from the main report and should not have a header or footer. A main report can contain many subreports. Subreports also may contain their own subreports if desired, but these must be defined within those subreports (not the main report). As a convention of the RBDMS developers, subreports are always named with the prefix “zz.”

You can embed a sub-report into the main report body by selecting the Sub-report control from the Toolbox pane in Visual Studio. When you insert a sub-report control on the main report body, you must specify the data link to the sub-report as a named parameter. For example, in MSRBDMS.NET, the sub-report **zzConstructCasing** is linked to the Well Completions report by the *ConstructKey* which is specified as the parameter value =Fields!ConstructKey.Value in the **Subreport Properties** dialog. If you forget to link the sub-report properly to the main report, the sub-report will not work.

## ***Recommended Reading***

Turley, Paul, Todd Bryant, James Counihan, and Dave DuVarney. *Professional SQL Server 2005 Reporting Services*. Programmer to Programmer series. Wiley Publishing, Inc. 2006. Chapters 4, 5, and 6.