User Guide

ActiveReports for .NET is a fully-integrated product which combines the features of the Visual Studio programming languages with user-friendly controls to provide a powerful report designer.

In the ActiveReports for .NET Documentation

- Introducing ActiveReports for .NET
- Getting Assistance
- Samples and Walkthroughs

Introducing ActiveReports for .NET

ActiveReports leverages the latest technologies including XML, scripting and CSS along with open architecture to provide you with a fully-integrated and user-friendly report designer.

This topic contains information about:

- ActiveReports Editions
- Data Dynamics Copyright Notice
- ActiveReports License Agreement
- Frequently Asked Questions
- Installation and Setup
- Upgrading Reports
- Architecture and Concepts
- Getting Started

ActiveReports Editions

- Standard Edition Features
- Professional Edition Features

Standard Edition Features

ActiveReports for .NET is a complete rewrite of the popular ActiveReports engine and report viewer. It includes the same power and flexibility of ActiveReports 2.0 and provides complete integration with the Visual Studio .NET Environment. ActiveReports for .NET Standard Edition supports the following features:

Designer

- Full integration with the .NET environment
- Familiar user interface
- Use with C# and VB.NET

- Compilation of reports included as part of the application for speed and security or included separately
- Designer hosting of .NET and user controls

Reporting Engine

- Managed code
- Binding to ADO.NET, XML and custom data sources
- Easy deployment with the report processing engine as a single assembly dll
- All of ActiveReports 2.0 features

Report Viewer

- Managed C# code
- Very small deployment assembly, suitable for use on the Internet
- Table of Contents/Bookmarks
- Hyperlinking

Export Filters

ActiveReports includes export filters to generate output into Rich Text Format (RTF) for word-processing, Portable Document Format (PDF), Microsoft Excel worksheets, HTML and DHTML for publishing your reports to the internet, TIFF for optical archiving and faxing and delimited text for spreadsheets and databases.

Professional Edition Features

ActiveReports for .NET is a complete rewrite of the popular ActiveReports engine and report viewer. It includes the same power and flexibility of ActiveReports 2.0 and provides complete integration with the Visual Studio .NET Environment. ActiveReports for .NET Professional Edition includes all of the features of the Standard Edition and supports the following additional features:

End-User Report Designer

The End-User Report Designer control is a run-time designer that may be distributed royalty free. It allows the ActiveReports designer to be hosted in an application and provides end-user report editing capabilities. The control's methods and properties provide easy access to save and load report layouts, monitor and control the design environment, and customize the look and feel to the needs of end users.

ASP.NET Integration

- Web server control provides convenience for running and exporting reports in ASP.NET.
- HTTP Handler extensions allow report files (RPX) or compiled assemblies containing reports to be dropped on the server and hyperlinked.

Web Viewer Control

- The Web Viewer control allows quick viewing of ActiveReports on the web as well as printing capability with the ActiveXViewer and AcrobatReader ViewerType properties.

HTTP Handlers

- The RPX HTTP Handler allows hyperlinking of an ActiveReport on a web page to return a HTML format or PDF format of the report for viewing and/or printing.
- Compiled Report HTTPHandler allows hyperlinking of an ActiveReport compiled in an assembly on a web page to a HTML format or PDF format of the report for viewing and/or printing.

these circumstances.

The software serial number and user must be registered with Data Dynamics in order to receive support or distribution rights.

You may not remove any proprietary notices, labels, and trademarks on the software or documentation.

You may not modify, de-compile, disassemble, reverse engineer or translate the software.

FILES THAT MAY BE DISTRIBUTED WITH YOUR APPLICATION:

Standard Edition License:

ACTIVEREPORTS.DLL, ACTIVEREPORTS.VIEWER.DLL ACTIVEREPORTS.INTEROP.DLL, AREXPORTEXCEL.DLL, ACTIVEREPORTS.HTMLEXPORT.DLL, ACTIVEREPORTS.PDFEXPORT.DLL, AREXPORTRTF.DLL, AREXPORTTEXT.DLL, AREXPORTTIFF.DLL

Professional Edition License:

ACTIVEREPORTS.DESIGN.DLL and ACTIVEREPORTS.WEB.DLL in addition to the Standard Edition files.

US GOVERNMENT RESTRICTED RIGHTS:

Use, duplication or disclosure by the United States Government is subject to restrictions as set forth under DFARS 252.227-7013 or in FARS 52.227-19 Commercial Computer Software - Restricted Rights.

TERM:

You may terminate your License and this Agreement at anytime by destroying all copies of the Product and Product Documentation. They will also terminate automatically if you fail to comply with any term or condition in this Agreement.

LIMITED WARRANTY:

This software and documentation are sold "as is" without any warranty as to their performance, merchantability or fitness for any particular purpose. The licensee assumes the entire risk as to the quality and performance of the software. Data Dynamics warrants that the media on which the Program is furnished will be free from any defects in materials. Exclusive remedy in the event of a defect is expressly limited to the replacement of media. In no event shall Data Dynamics or anyone else who has been involved in the creation, development, production, or delivery of this software be liable for any direct, incidental or consequential damages, such as, but not limited to, loss of anticipated profits, benefits, use, or data resulting from the use of this software, or arising out of any breach of warranty.

Frequently Asked Questions

Is ActiveReports 100% managed?

ActiveReports includes the following distributable DLLs:

Managed DLLs:

- ActiveReports.DLL - This is the reporting engine. Managed and written in C#.
- ActiveReports.Design.DLL - This is the run-time end user report designer. Managed and written in C#.
- ActiveReports.Viewer.DLL - This is the Windows Forms viewer. Managed and written in C#.
- ActiveReports.Web.DLL - This is the Web Forms viewer and RPX handler. Managed and written in C#.
- ActiveReports.PdfExport.DLL - This is the PDF export filter. Managed and written in C#.
- ActiveReports.HtmlExport.DLL - This is the HTML export filter. Managed and written in C#.

Unmanaged DLLs:

- ActiveReports.Interop.DLL - This is the auxiliary COM interop DLL. Unsafe C# code for OLE object hosting and other COM functions.
- ARExport*.DLL - These are the export filters. Managed wrappers around unmanaged VC++ code.

Note The export DLLs will be converted to managed C# after the first release. The above are the only distributable DLLs. The core components are all managed and use all of what .NET has to offer.

ActiveReports also includes the following non-distributable DLLs:

- ARDBWizard.DLL - This is the design-time Report Wizard. Managed C#.
- ARTaskPane.DLL - This is the design-time wizards container. Managed C#.
- ARVSPackage.DLL - This is the VS Integration package. Unmanaged VC++ (Since integration with VS requires COM, this cannot be done any other way).

Why is the viewer icon grayed out?

The icon may be grayed out for 2 reasons:

1. The control selected in the components list was for the wrong viewer control. ActiveReports has two separate viewer controls. One is DataDynamics.ActiveReports.WebViewer, which is a viewer control that only works on Web Forms, and the other is DataDynamics.ActiveReports.Viewer, which is a viewer control that works only on Windows Forms.
2. The correct form is not selected. If a Windows Form or Web Form is not selected in the project, the viewer icon may be grayed out. Verify the correct viewer component is selected for the project:

   For the Windows Form Viewer - DataDynamics.ActiveReports.Viewer - ActiveReports.Viewer(x.x.x.xxxx)

   For the WebForm Viewer - DataDynamics.ActiveReports.Web - ActiveReports.WebViewer(x.x.x.xxx)

Why am I getting an ambiguous reference error?

This error will occur if the report's class file has "Imports System.Windows.Form" or "using System.Windows.Form;." The reason for the error is the Windows.Forms namespace and the ActiveReports namespace have definitions for Textbox, Label etc. In order to prevent the error, the code will need to use the full System.Windows.Form namespace when referencing Windows.Form objects.

Installation and Setup

This topic will help you find out what is necessary to install ActiveReports and direct you to appropriate help for the installation process.

- Requirements
- Installed Files
- Troubleshooting Installation
- Verifying Package Installation

  Tip Close Visual Studio .NET before running the installation program to allow the help files to be integrated into Visual Studios common help collection.

Hardware and Software Requirements

This topic includes hardware and software requirements for installing and using ActiveReports for .NET.

Hardware requirements (minimum)

Processor: PC with a Pentium II-class processor 450 MHz
Operating System: Windows® 2000, Windows XP or Windows NT 4.0

Software requirements

Microsoft .NET framework
Microsoft Visual Studio .NET

Installed Files

Folder/Files	Description
<Common Files>\Data Dynamics\ActiveReports for .NET	
ActiveReports.DLL	Run-time engine assembly file
ActiveReports.Design.DLL	Designer assembly file
ActiveReports.Interop.DLL	Native functions assembly file
ActiveReports.Viewer.DLL	Viewer assembly file
ActiveReports.Web.DLL	Web assembly file
ActiveReports.Web.Design.DLL	Web designer assembly file
ARVSPackage.DLL	Visual Studio .NET Integration Package
ActiveReports.HtmlExport.DLL	HTML Export DLL
ActiveReports.PdfExport.DLL	PDF Export DLL
ARExportTIFF.DLL	TIFF Export DLL
ARExportExcel.DLL	Excel Export DLL
ARExportRTF.DLL	RTF Export DLL

ARExportText.DLL	Text Export DLL
ARCol.Hx*	ActiveReports Help Integration Collection
ddARRef.HxS	ActiveReports Help File--Class Library
ddARUG.HxS	ActiveReports Help File--User Guide
License.rtf	Data Dynamics ActiveReports License Agreement and Limited Warranty
<Common Files>\Data Dynamics\ActiveReports for .NET\1033	
ARVSPackageUI.DLL	Visual Studio .NET integration DLL localized UI
<Common Files>\Data Dynamics\ActiveReports for .NET\Wizards	
ARAccessWizard.DLL	Access to ActiveReports wizard
ARDBWizard.DLL	New Report Wizard Add-In
ARInstallExt.DLL	ActiveReports installation auxiliary file
arinstallext.InstallState	ActiveReports installation auxiliary file
DDAccessHelper.dll	Access wizard auxiliary file
WebKey.exe	Web.config key generator
ARTaskPane.DLL	Report Wizards IDE Task Pane
<Application Folder>	
Data\Nwind.mdb	Samples database file
Data\streamSample.mdb	Streaming sample database file
Deployment\ActiveReportsDistrib.msm	ActiveReports merge module file
Deployment\arview2.cab	ActiveX viewer cab file
Introduction\iddlogo.gif	Readme image file
Introduction\itopimage1.gif	Readme image file
Introduction\itopimage2.gif	Readme image file
Introduction\assemblies.gif	Readme image file
Introduction\readme.html	ActiveReports Readme file
Samples\samples.txt	Samples description text file
Samples\CSharp\*.*	C# sample projects
Samples\VB\*.*	Visual Basic sample projects

Troubleshooting Installation

Why can't other users access or use ActiveReports on my machine?

The installation for ActiveReports for .NET gives the user the option to install the program for everyone or the current user. If it is installed only for the current user, other users on the same machine will not be able to access it or use it.

I just installed ActiveReports for .NET. Why can't I see the help files?

If the installation was run while Visual Studio was open, the help files will not be integrated until the user exits Visual Studio and reopens it.

Why do I get, "The installer was interrupted before Data Dynamics ActiveReports for .NET... could be installed. You need to restart the installer to try again" when I run the ActiveReports Setup?

The most likely cause for this error is a permissions issue. Either the folder the setup is pointing to or the folder containing the setup files does not have the correct permissions. The user needs to verify the system account for the local machine has permissions to the folder containing the setup and verify the user installing the program has the appropriate permissions.

Verifying Package Installation

To verify package installation

1. Open Visual Studio .NET.
2. You should see the ActiveReports logo on the splash screen.
3. Open **Help** > **About Dialog** and verify the "Data Dynamics ActiveReports" entry in the installed products list.

Upgrading Reports

ActiveReports allows you to upgrade your reports from other versions of ActiveReports and other report programs.

- Changes Between ActiveReports 2.0 and ActiveReports for .NET
- Converting Microsoft Access Reports
- Migrating From ActiveReports 2.0

Changes Between ActiveReports 2.0 and ActiveReports for .NET

Report Classes

Data Dynamics attempted to keep to a minimum the number of changes to the report object model from ActiveReports 2.0. Most of the changes are due to class refactoring and renaming of objects and members to closely match the .NET Framework naming conventions. Listed below are the significant changes in the object model.

- ActiveReport.Show removed: ActiveReports class is no longer a Window class. This requires using the viewer control to preview the output of a report.
- Strong-typed Section classes: ActiveReports for .NET includes classes for each of the section types with their own unique properties. The old Section object is still available and the SectionCollection class holds items of the Section type. The new section classes are Detail, GroupHeader, GroupFooter, PageHeader, PageFooter, ReportHeader and ReportFooter.
- New Stylesheet class: Provides access to the styles defined in the report and allows you to change the individual style item properties.
- Image control renamed to Picture.
- Field control renamed to Textbox.
- Supported justified text alignment option for textboxes and labels.
- Indirect support of ActiveX controls through .NET wrappers and the new CustomControl class.
- Three added data source classes which replace the data controls: OleDbDataSource, XmlDataSource and SqlClientDataSource.
- Split Pages collection: the Pages class from ActiveReports 2.0 is refactored into a PagesCollection class and a Document class. The new Document class has all the members to save/load RDF files and streams.
- Changed measurements from twips to inches.

Printing

- Added PrintController and PrintControllerWithStatus classes to make the printing model similar to the .NET Framework.
- Use of the .NET Framework Printer and PrinterSettings classes by the viewer control. An optional unsafe printer class is also included for advanced printing and print job control similar to the ActiveReports 2.0 class.

Viewer

- A rewritten report viewer control to take full advantage of the .NET framework classes.
- Complete revision of the Toolbar and Tools classes.
- Separation of the Table of Contents tree control from the TOC collection (renamed to BookmarksCollection).
- No binding of the viewer control to an ActiveReport object. Instead, it binds to a Document object.

Converting Microsoft Access Reports

Access reports can be easily converted to ActiveReports format by running the Access upsizer wizard. Due to differences between products, the extent to which your reports will be converted will depend on your specific report layout. However, since Data Dynamics provides source code, you can modify the resulting ActiveReport to achieve the results you desire. To launch the upsizer, open a project in Visual Studio, click on **Tools > ActiveReports Wizard**, then click on **Access Import**. This launches the Access to ActiveReports Wizard.

Migrating from ActiveReports 2.0

ActiveReports for .NET can use existing ActiveReports 2.0 report layout files (RPX) after some modifications to the scripting code. ActiveReports 2.0 designer files (DSR/DSX) must be saved as RPX files in the ActiveReports 2.0 Designer before they can be imported into ActiveReports. Since ActiveReports does not import any Visual Basic or scripting code into .NET, the code will need to be rewritten using the appropriate language in the new .NET environment.

Architecture and Concepts

This topic will introduce you to the basic structure and concepts of ActiveReports for .NET to enable efficient report creation.

- Events
- Layout Files
- Parameters
- Report and Page Settings
- Report Execution
- Report Structure
- Scripting

Events

In a report, regardless of the type or content of the various sections, there are three events for each section: Format, BeforePrint and AfterPrint.

Because there are many possible report designs, the event-firing sequence must be dynamic in order to accommodate individual report demands.

Out of the three events, the Format event generally is used the most often, followed by the BeforePrint event and, in rare circumstances, the AfterPrint event.

Format event

This event fires after the data is loaded and bound to the controls contained in a section, but before the section is rendered to a page.

The format event is the only event where the section's height may be changed. This section may be used to set or change the properties of any controls or load subreport controls with subreports.

If the CanGrow or CanShrink property of any control contained with a section, or the section itself, is set to true, all of the growing and shrinking of controls contained in this section, and the section itself, takes place in the Format event. Because of this, information about a control or a section's height cannot be obtained in this event.

BeforePrint event

This event fires before the section is rendered to the page.

The growing and shrinking of the section and all controls contained in a section have already taken place by the time this event fires. Use this section to resize any controls if needed.

Since all controls and section growth have already taken place by the time this event fires, this event may be used to get an accurate height of the section, or, if needed, any controls contained in it. Any controls in the BeforePrint event may be resized but not the height of the section itself.

AfterPrint event

This event fires after the section is rendered to the page.

Although AfterPrint was an important event prior to ActiveReports Version 1 Service Pack 3, it is rarely used in any of the newer builds of ActiveReports. When you place code in the section events, you likely will place your code in either the Format event or the BeforePrint event. This event is still useful for drawing on the page after text has already been rendered to the page.

Layout Files

Report layouts in ActiveReports are automatically saved as RPX files. This is an XML-formatted file which contains the layout information and can contain the scripts of the report. RPX files using scripting allow distributed reports to be changed and modified without having to recompile the project. They also make it possible to use a database of report file names to set up a collection of reports to run. An RPX file using scripting also can be used as a stand-alone file in a web project or a stand-alone file for the HTTP handler.

Parameters

Parameters and Simple Reports

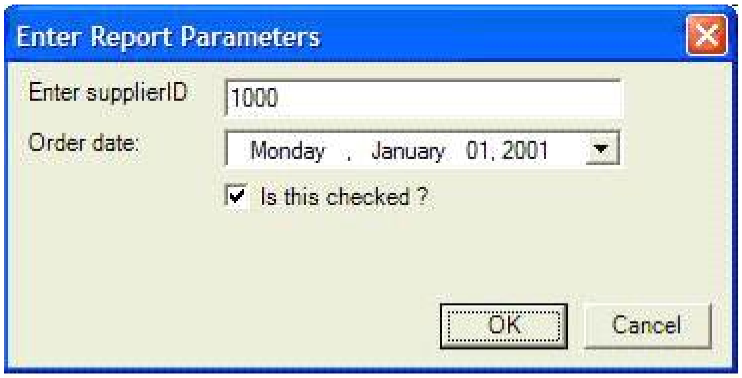
The Parameters dialog can be used to prompt the user for input when reports are generated. If you add <%FieldName | PromptString | DefaultValue | Type%> to the reports SQL string, it will cause the Parameters dialog to be displayed.

The Field name is the name of the field you wish to request (e.g. CustomerID or LastName). The Prompt string is a string value indicating text that will appear in the dialog next to the control (e.g. Enter Customer ID:). Setting the default value will automatically set a default value. For example, if you have a report that generates based on a date, you can have the default for the field set to the current date so users can just hit "Enter", unless they want to generate a report based on a new date. Type indicates what type of data will be requested.

The values can be: nothing(string), S for string, D for date, B for Boolean. A string type will give a textbox for input, a D type will give a calendar drop-down control for input and a B type will give a checkbox for input.

> **Note** For Strings: If you specify a default value that is enclosed in single or double quotes, it will be recognized and will output the same quotes to SQL when replacing. For Booleans : if you specify true/false for DefaultValue it will generate true/false for SQL output. If you specify 0,1, it will output 0 or 1.

Example: "SELECT * FROM products INNER JOIN categories ON products.categoryid = categories.categoryid WHERE products.supplierID =<%SupplierID|Enter supplierID|1000%> and OrderDate=#<%Date|Order date:|1/1/2001|D%># and Discount=<%bool| Is this checked ?|true|B%>"


Enter Report Parameters
Enter supplierID: 1000
Order date: Monday , January 01, 2001
[✓] Is this checked ?

Note The FieldName is the only required parameter; the rest are optional.

Parameters and Subreports

Parameters can be used with subreports to connect the subreport to the parent report. By setting a parameter for the field that links the parent report to the child subreport, the parent report can pass the information to the child through the parameters. The main differences when working with subreports and parameters are:

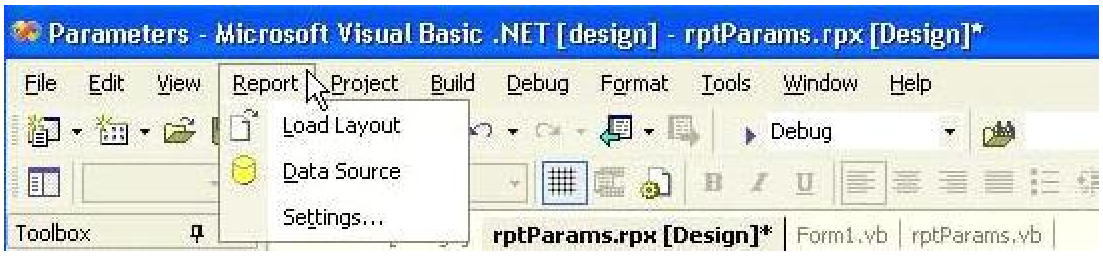
- The subreports *ShowParametersUI* should be set to False.
- The subreports SQL query should be set to use the parameter syntax = <%fieldname%>.

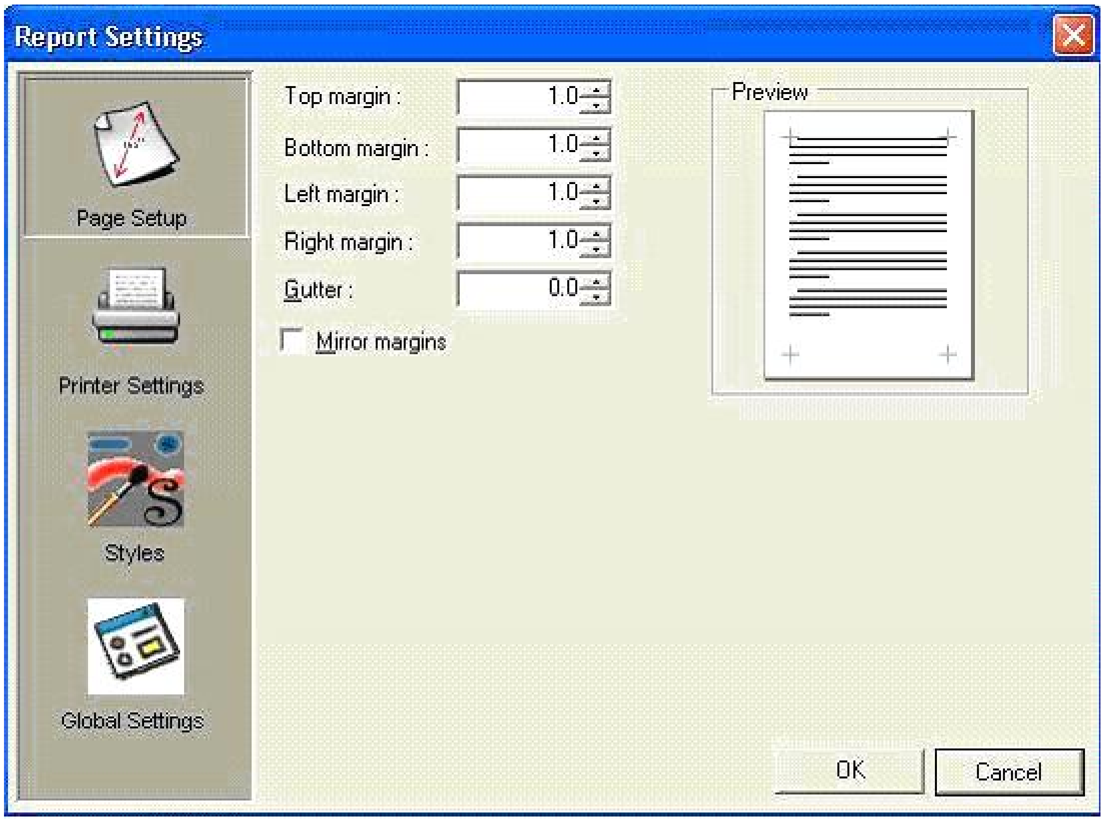
  Note Both report queries must contain the same field (so the main report must have a categoryID field and the subreport also must have a categoryID field.

Report and Page Settings

The Page (Report) Setup Dialog

With ActiveReports, page setup in your report can be modified at design time, as well as at run time. The Page Setup dialog can be accessed by selecting **Report** > **Settings...** from the toolbar menu.


Parameters - Microsoft Visual Basic .NET [design] - rptParams.rpx [Design]*
File Edit View Report Project Build Debug Format Tools Window Help
Load Layout
Data Source
Settings...
Debug
Toolbox
rptParams.rpx [Design]*
Form1.vb
rptParams.vb



Top margin : 1.0
Bottom margin : 1.0
Left margin : 1.0
Right margin : 1.0
Gutter : 0.0
Mirror margins

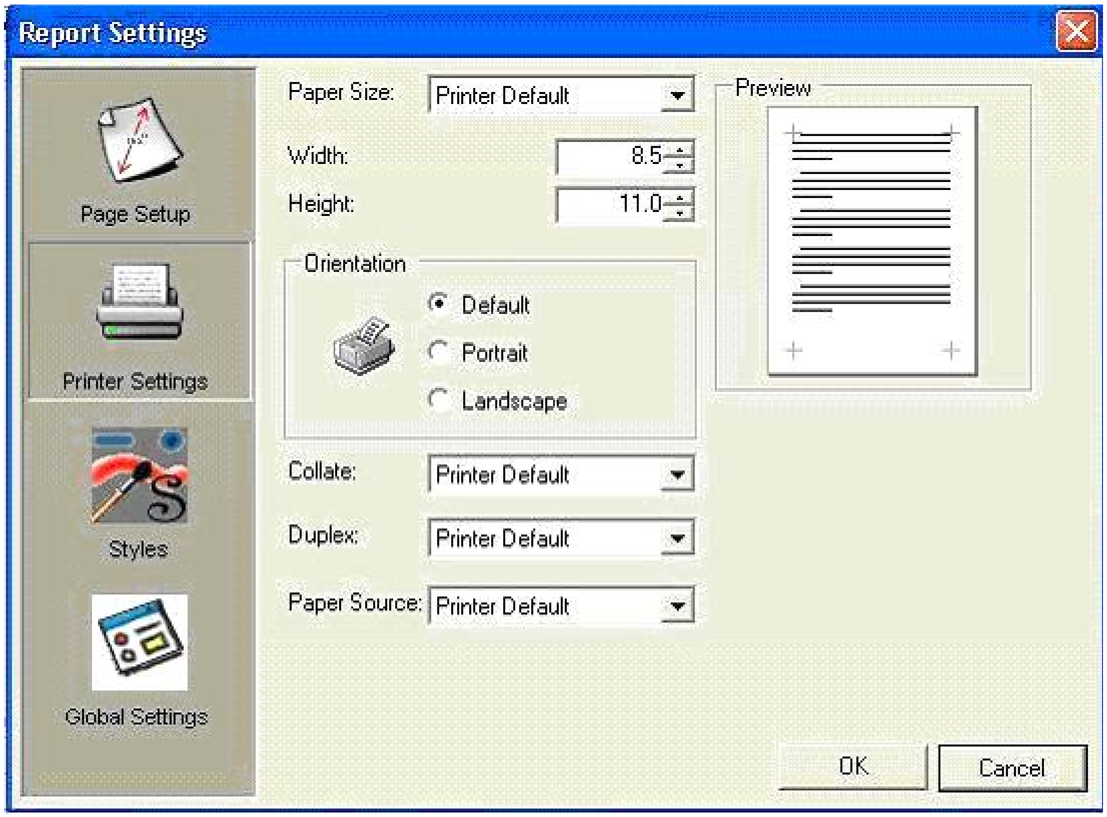
From the Page Setup dialog, changes can be made to the report margins (left, right, top and bottom), a gutter can be specified and the Mirror margins option can be selected.

By setting a gutter and selecting Mirror margins, reports can be set up easily for publishing purposes. When Mirror margins is selected, the inner margins in the report are set for opposite pages to be the same width and the outside margins for opposite pages to be the same width.

Specifying a gutter gives extra space between the edge of the page and the margins. This allows reports to be bound together.

The Printer Settings Dialog

With ActiveReports, printer settings can be modified at design time, as well as run time. The Print Settings dialog (shown below) can be accessed by selecting **Report** > **Settings...** from the toolbar menu and then selecting the Printer Settings option button from the Report Settings dialog box.


Report Settings

Paper Size: Printer Default

Width: 8.5

Height: 11.0

Orientation: [✓ Default] Portrait Landscape

Collate: Printer Default

Duplex: Printer Default

Paper Source: Printer Default

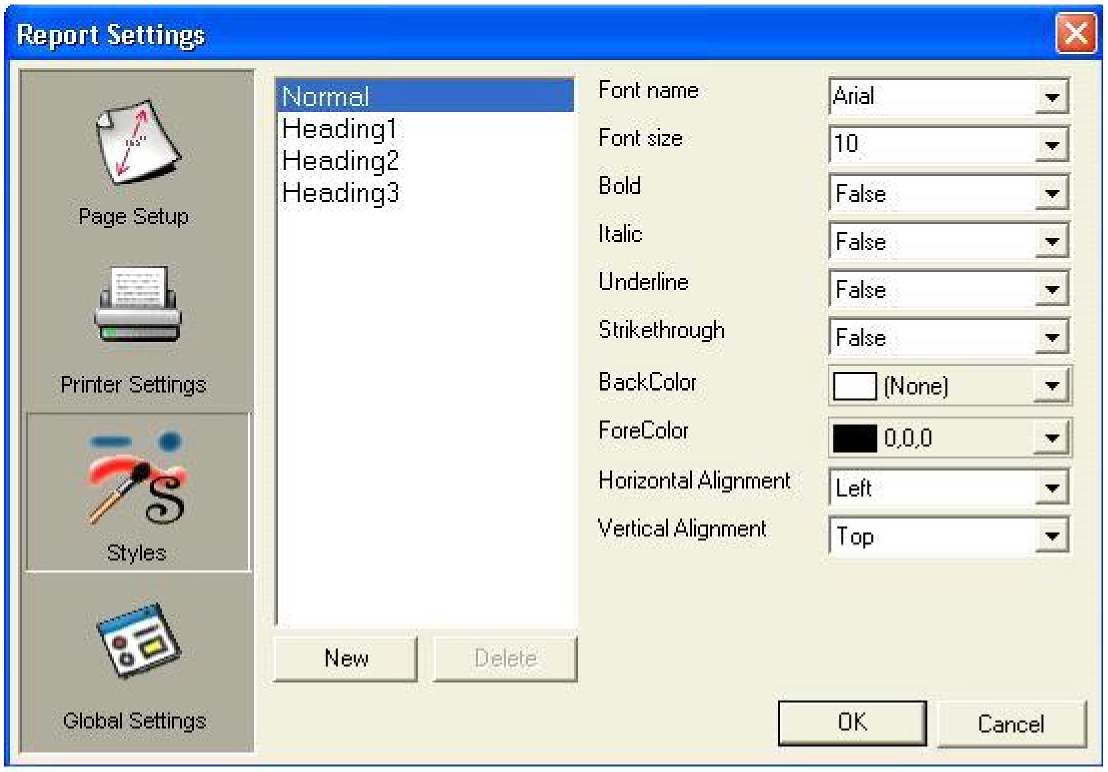
From the Printer Settings dialog, changes can be made to the printer paper size and orientation.

A custom paper size can be set by selecting Custom paper size from the Paper size drop down box. Once this option has been selected, the width and height options will allow a specific height and width to be set.

The Printer Settings dialog also lets the user choose the type of collation to use, whether or not the report should be duplexed and the location of the paper source.

The Styles Dialog

With ActiveReports, style sheet settings can be created and/or applied. The Styles dialog (shown below) can be accessed by selecting **Report** > **Settings...** from the toolbar menu and then selecting the Styles option button from the Report Settings dialog box.



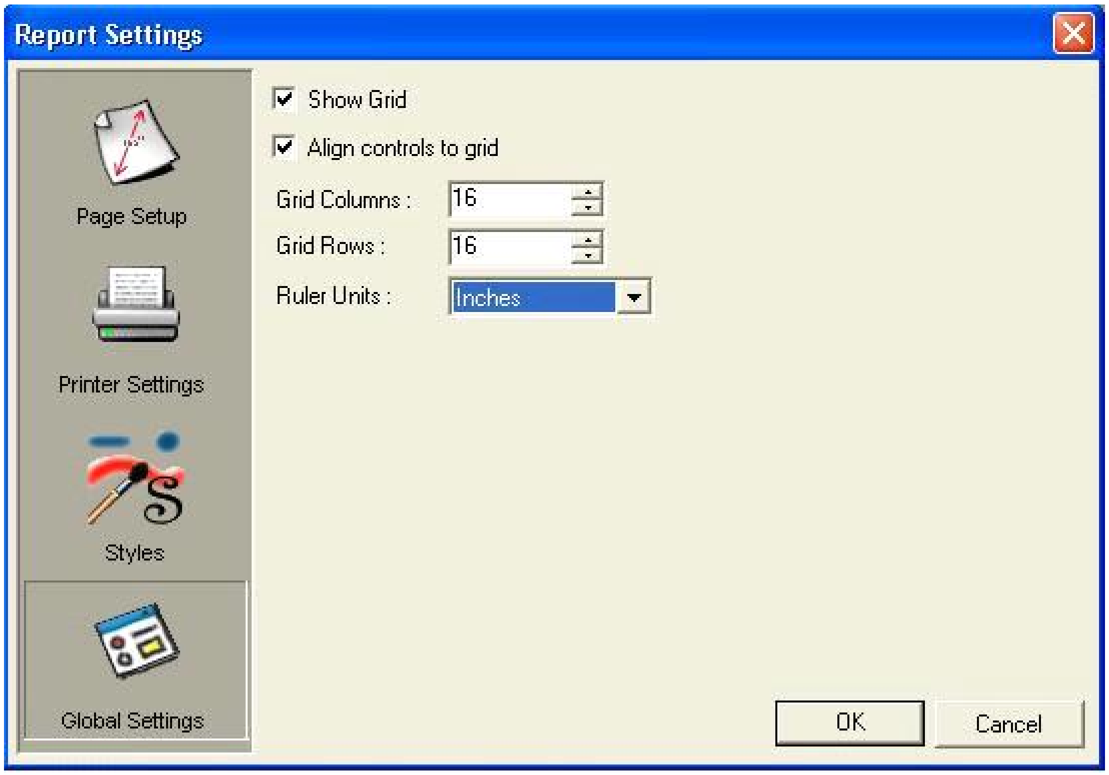
Normal
Heading1
Heading2
Heading3

Font name	Arial
Font size	10
Bold	False
Italic	False
Underline	False
Strikethrough	False
BackColor	(None)
ForeColor	■ 0,0,0
Horizontal Alignment	Left
Vertical Alignment	Top

From the Styles dialog, changes can be made to the appearance of text associated with controls, either by applying an existing style sheet, creating and applying a new style sheet or by modifying and applying an existing style.

The Global Settings Dialog

With ActiveReports, global report settings can be modified at design time. The Global Settings dialog (shown below) can be accessed by selecting **Report** > **Settings...** from the toolbar menu and then selecting the Global Settings option button from the Report Settings dialog box.



☑ Show Grid

☑ Align controls to grid

Grid Columns : 16

Grid Rows : 16

Ruler Units : Inches

From the Global Settings dialog, changes can be made to the design surface, including showing or hiding the grid, setting the controls to align to the grid, setting the number of column or rows on the grid and changing the ruler units to inches or centimeters.

Report Execution

ActiveReports report execution begins by raising the ReportStart event. At this point, accessing data source properties might cause DataInitialize to fire. The report validates any changes made to the report structure in ReportStart.

Printer settings are applied next.

If DataInitialize is not fired during the ReportStart event, it will be fired. The data source will be opened. If there are any parameters in the data source with unset values and "ShowParameterUI" is set to True, ActiveReports displays a parameters dialog and fires "ParameterUIClosed" when the dialog is closed. If the report is a subreport and requires parameters, ActiveReports binds the subreport parameters to any fields in the parent report.

Next, the FetchData event fires. If there is no data, the NoData event is raised.

Group sections are bound and sections begin rendering on pages.

Events are then fired for processing the report header, followed by page header, groups, detail and page footer for each page in the report. The cancel flag is checked after each event.

The speed in processing and output generation of ActiveReports is attributed to its intelligent, multi-threaded, single-pass processing. ActiveReports will process and render each page as soon as the page is ready. If ActiveReports is not able to fully render a page because of unknown data elements or because the layout is not final, it places the page in cache until the data is available.

Summary fields and KeepTogether constraints are two reasons that a page might not be rendered completely. The summary field is not complete until all the data needed for calculation is read from the data source. When a summary field such as a grand total is placed ahead of its completion level, such as in the report header, the report header and all following sections will be delayed until all of the data is read.

The KeepTogether property determines whether a section should print in its entirety on the same page. When this property is set to True, the section will print on the same page without any page breaks. A False setting allows the section to be split across two or more pages. If the KeepTogether property is set to True, but the section is too large for the current page, or to fit fully on the next page, the KeepTogether property will be ignored.

The GroupKeepTogether property determines whether group header and footer sections will print as a single block on the same page. The property defaults to None which allows the group block to be split across pages. When you set this property to All, ActiveReports attempts to print the complete block on the same page without any page breaks. When a complete block does not fit on a single page, it will be split across two or more pages. The third option, FirstDetail, prevents any widowed group header sections. The group header will always print with at least one detail section.

Report Structure

A report section contains a group of controls that are processed and printed at the same time as a single unit. ActiveReports defines the following section types:

Report Header

A report can have one report header section that prints at the beginning of the report. This section generally is used to print a report title, a summary table, a chart or any information that needs only to appear once at the report's start.

Report Footer

A report can have one report footer section that prints at the end of the report. This section is used to print a summary of the report, grand totals or any information that needs to print once at the report's end.

Page Header

A report can have one page header section that prints at the top of each page. It is the first section that prints on the page except when the page contains a report header section. The page header section is used to print column headers, page numbers, a page title or any information that needs to appear at the top of each page in the report.

> **Note**   It is not recommended to bind controls to a page header as results may be unpredictable.

Page Footer

A report can have one page footer section that prints at the bottom of each page. It is used to print page totals, page numbers or any other information that needs to appear at the bottom of each page.

Group Header/Footer

A report can consist of single or multiple nested groups, with each group having its own header and footer sections. The header section is inserted and printed immediately before the detail section. The footer section is inserted and printed immediately after the detail section.

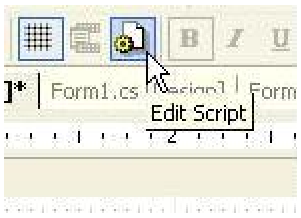
Detail

A report has one detail section. The detail section is the body of the report and one instance of the section is created for each record in the report.

Scripting

ActiveReports allows you to use scripting to provide ease in reporting functionality. Scripting permits reports saved to an RPX file to contain code. This characteristic allows the options of stand-alone reporting and web reporting without requiring .vb or .cs files. By including scripting when the report is saved as an RPX file, it can later by loaded, run and displayed directly to the viewer control without using the designer. Scripting can also be used in conjunction with RPX files to allow distributed reports to be updated without recompiling.

Scripting can be used by adding C# code to the script editor at design time or by using *rpt.Script* at run time. The script is then saved to the RPX file.



The AddNamedItem and AddCode methods are used to add items to the reports script. By using *AddNamedItem* or *AddCode*, code elements from inside the .NET project can be used inside the scripts. By using *AddNamedItem*, scripts can become aware of functions in a class contained in the .NET project. By using *AddCode*, actual code segments can be added to the script at run time. Since the RPX file can be read with any text editor, *AddCode* or *AddNamedItem* can be used to add secure information to a project, such as a connection string.

AddScriptReference can be used to add an assembly reference to the script. This will allow users to add a reference in the script to access assemblies in their projects. *AddScriptReference* is only needed if the script accesses assemblies that are not already initialized in the project. For example, to access "System.Data.DataSet" inside the script, you would need to add a reference by calling "rpt.AddScriptReference("System.Data.Dll")".

Scripting Concepts to Remember:

- Controls referenced inside the script must be public
- If the RPX file does not have an associated codebehind file, you will need to refer to the controls and sections by calling "rpt.Sections[<sectionname>]" or "rpt.Sections[<sectionname>].Controls[<controlname>]"
- The report instance is referred to in the report as "rpt." This is similar to "me" and "this" in the codebehind files. You must use "rpt" to gain access to the report and its controls
- The report class has to be public for scripting to access public methods and/or functions (this is done by default)

Getting Started

This topic will show you how to begin using ActiveReports by explaining different aspects of ActiveReports and showing you how to include it in your Visual Studio .NET IDE (Integrated Development Environment).

- ActiveReports Designer
- Adding ActiveReports Controls to the Visual Studio Toolbox
- Adding an ActiveReport to a Visual Studio .NET Project
- Binding Reports to a Data Source
- Grouping Data
- Licensing Applications
- Localizing the Viewer Control
- Manually Configuring Web Samples
- Metric Units
- Saving and Loading RDF Files
- Saving and Loading RPX Files

ActiveReports Designer

With its various tools and qualities, ActiveReports for .NET offers great flexibility in constructing report projects. In this section, you will learn how to use the different features of the ActiveReports Designer.

- ActiveReports WinForm Viewer
- Adding a Report to your Project
- Design Surface
- Loading an Existing Report Layout
- Report Menu
- Toolbars
- Toolbox

ActiveReports WinForm Viewer

To use the ActiveReports WinForm Viewer to preview report output

1. Add an ActiveReport to your Visual Studio project and rename it rptMain.
2. Add a new "Windows Form" to your project.
3. Click on the ActiveReports viewer control in the appropriate toolbox and drag it onto Form1.
4. Set the viewer control's Dock property to Fill.

To write the code for the viewer in Visual Basic

Right-click on Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

- Format the viewer to show the report when it is run

To write the code for the viewer in C#

Click on the blue section at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for Form1. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

- Format the viewer to show the report when it is run

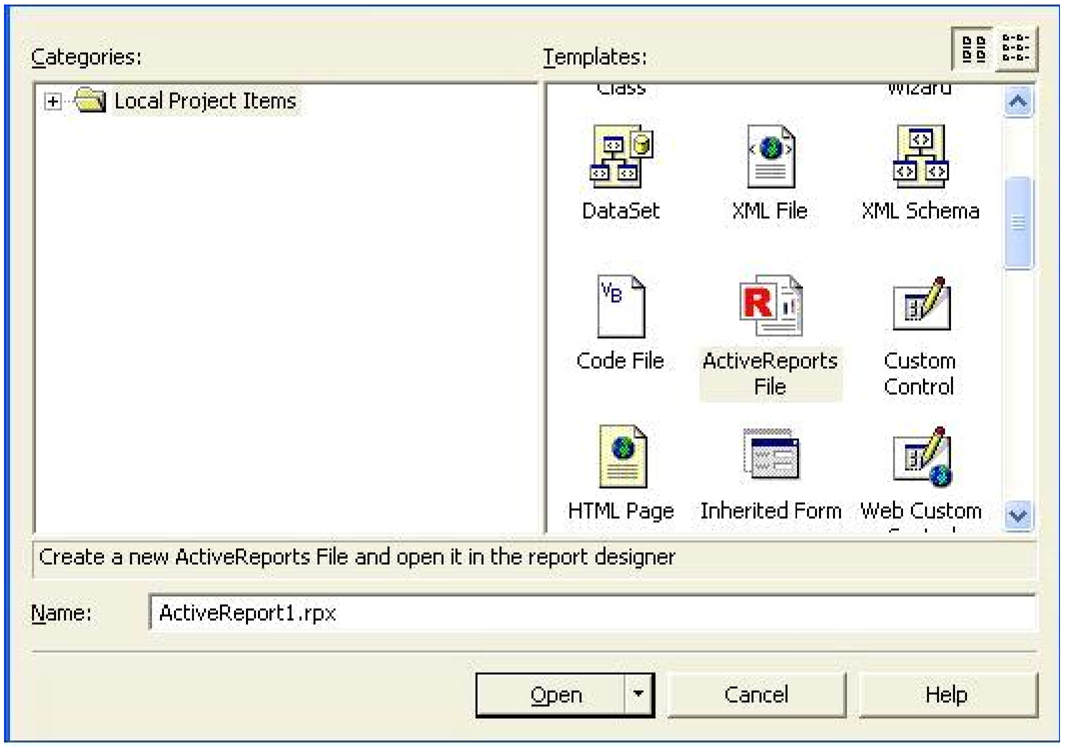
The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles MyBase.Load
         Dim rpt As New rptMain()
         Viewer1.Document = rpt.Document
         rpt.Run()
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         rptMain rpt = new rptMain();
         this.viewer1.Document = rpt.Document;
         rpt.Run();
}
```

Adding a Report to your Project

To add a report to your project

1. Open a new or existing project.
2. Click on **Project** > **Add New Item** (Ctrl+Shft+A)
3. Select **ActiveReports File** and rename the file.



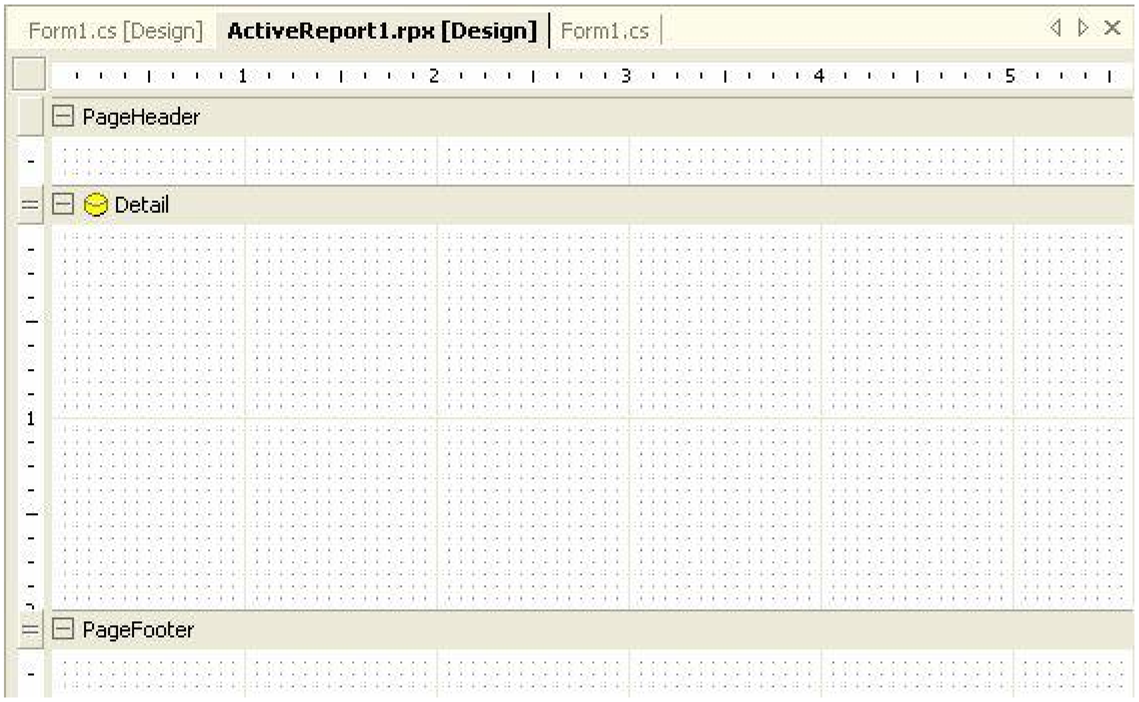
Create a new ActiveReports File and open it in the report designer

Name: ActiveReport1.rpx

4. Click **Open** to add the report to your project.

Design Surface

The ActiveReports design surface leverages your current knowledge of Visual Studio .NET's designer interface and provides full integration within the Visual Studio environment.



To access the ActiveReports design surface

1. Open a Visual Studio project.
2. Add an ActiveReport to your project.
3. Once the report is added, you will see the report design surface.

The default ActiveReports design surface is made up of the following base components:

PageHeader--This section can be used to print column headers, page numbers, page titles, or any information that needs to be printed once at the top of each page.

Detail section--This section is the body of the report that prints once for each record in the data source.

DataSource control icon--This control can be used to connect the report's data source settings to an existing data source.

PageFooter--This section can be used to print page totals, page numbers or any other information that needs to be printed once at the bottom of each page.

Loading an Existing Report Layout

To load an existing Report Layout

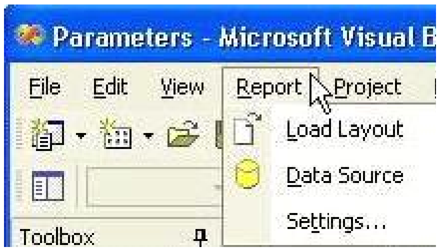
1. Open a Visual Studio project.
2. Click on **File > Open > File...**
3. Select the RPX report layout from from the appropriate location.
4. Click **Open** to load the report layout.
   -or-
5. While in an ActiveReport, click **Report > Load Layout**.
6. Select the RPX file from the appropriate location.
7. Click **Open** to load the selected report layout.

Report Menu

The report menu allows access to load report layouts, modify the report data source or modify report settings.

To access the report menu

1. Open a new or existing ActiveReport.
2. Click on any section in the report to select it.
3. Click on **Report** from the main toolbar.



The Report Menu allows the following options:

- **Load Layout**--the Load Layout option allows access to load an existing report layout into the open ActiveReport.
- **Data Source**--the Data Source option allows access to add Data Source parameters or modify existing Data Source settings.
- **Settings**--the Settings option allows access to change printer or page settings, style sheets or global settings.

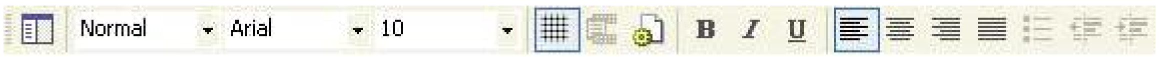
Toolbars

The toolbars in ActiveReports can be easily customized. ActiveReports' toolbars allow developers to rearrange buttons and menu options, as well as hide, display, dock or float toolbars.

To access a toolbar's context menu

- Right-click anywhere in the toolbar area

The context menu allows you to show or hide toolbars by selecting the toolbar name from the menu. In addition, you can customize the toolbars or create a new toolbar from the customize option on the menu.

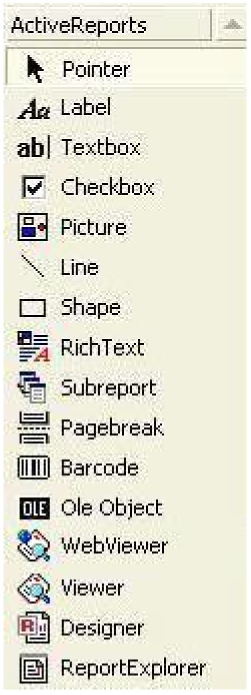


The ActiveReports toolbar is made up of the following components:

- *Report Explorer*--Shows or hides the report explorer tree and the fields list
- *Style Sheets*--Sets the style sheet for a control
- *Font*--Sets the typeface of the selected label, checkbox or textbox control
- *Size*--Sets the font size of the selected label, checkbox or textbox control
- *View Grid*--Turns the grid display on or off
- *Reorder Groups*--Displays the groups order dialog
- *Edit Script*--Starts ActiveReports Script Editor
- *Bold*--Sets the bold typeface on or off
- *Italic*--Sets the italic typeface on or off
- *Underline*--Sets the underline typeface on or off
- *Align Left*--Aligns the text left in the control area
- *Align Center*--Aligns the text centered in the control area
- *Align Right*--Aligns the text right in the control area
- *Justify*--Justifies the text in the control area
- *Bullets*--Adds bullets to the text in the RichText control area
- *Decrease Indent*--Decreases the indent of the text in the RichText control area
- *Increase Indent*--Increases the indent of the text in the RichText control area

Toolbox

The ActiveReports toolbox displays a variety of controls available for use in ActiveReports. The items available from the toolbox change depending on the designer currently in use.



To access the ActiveReports toolbox

1. Open a Visual Studio project.
2. Add an ActiveReport to the project.
3. Click on **View** > **Toolbox**.
4. Click on the *ActiveReports* tab.

The ActiveReports toolbox is made up of the following components:

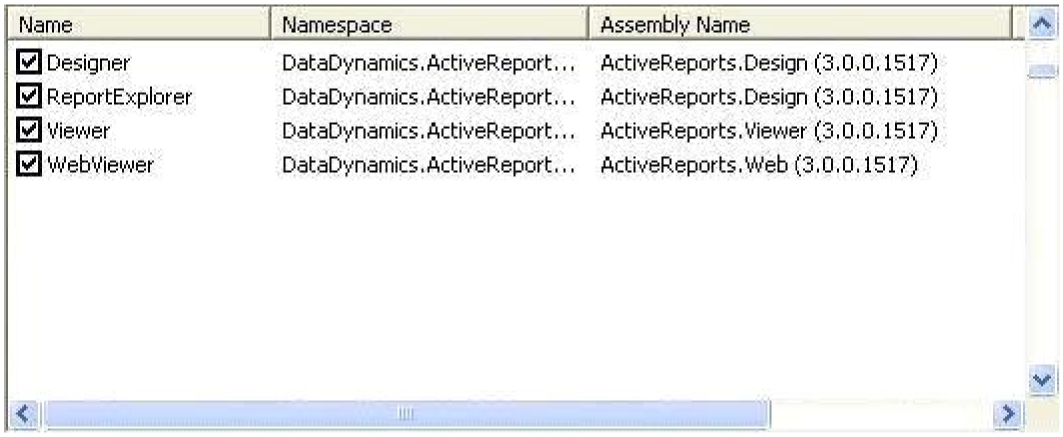
- *Pointer*--Allows you to select controls or sections of the report
- *Label*--Allows you to insert a new static label control
- *Textbox*--Allows you to insert a text box, bound to a database field or unbound
- *Checkbox*--Allows you to insert a check box, bound to a database field or unbound
- *Picture*--Allows you to insert an image, loaded from a file
- *Line*--Allows you to insert a line control
- *Shape*--Allows you to insert a rectangle, circle or square shape
- *RichText*--Allows you to insert an ActiveReports RichText control
- *Subreport*--Allows you to insert a Subreport control to link to another report
- *PageBreak*--Allows you to insert a page break within a selection
- *Barcode*--Allows you to insert an ActiveReports Barcode control
- *Ole object*--Allows you to insert an OLE object, bound to a database field, or unbound
- *WebViewer*--Allows you to insert a control to view an ActiveReport on the web
- *Viewer*--Allows you to insert an ActiveReports Viewer control
- *Designer*--Allows you to insert an ActiveReports Designer control
- *ReportExplorer*--Allows you to insert an ActiveReports ReportExplorer control

Adding ActiveReports Controls to the Visual Studio Toolbox

Adding the ActiveReports Controls

To add the controls

1. Right-click on the toolbox tab where you want to add ActiveReports controls.
2. Select **Customize Toolbox**.
3. Select ".NET Framework Components" tab.
4. Select the Designer, ReportExplorer, Viewer and WebViewer controls in DataDynamics.ActiveReports namespaces in the components list view.



Name	Namespace	Assembly Name
☑ Designer	DataDynamics.ActiveReport...	ActiveReports.Design (3.0.0.1517)
☑ ReportExplorer	DataDynamics.ActiveReport...	ActiveReports.Design (3.0.0.1517)
☑ Viewer	DataDynamics.ActiveReport...	ActiveReports.Viewer (3.0.0.1517)
☑ WebViewer	DataDynamics.ActiveReport...	ActiveReports.Web (3.0.0.1517)

5. Click **OK** to add the controls to your selected toolbox.

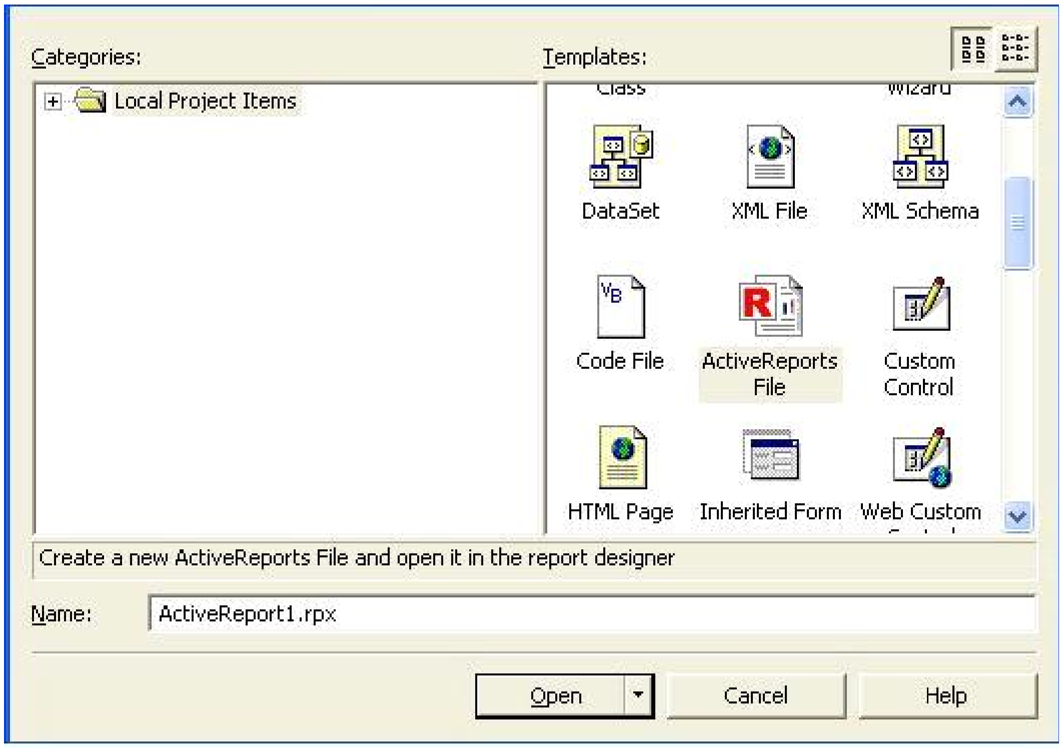
Adding An ActiveReport to a Visual Studio .NET Project

Creating the Project

The first step is creating a new Visual Studio Project.

To create the project

1. From the **File** menu, click on **New** and choose **Project.**
2. Select Project, then Add New Item.
3. Select **ActiveReports File** and name your new report.

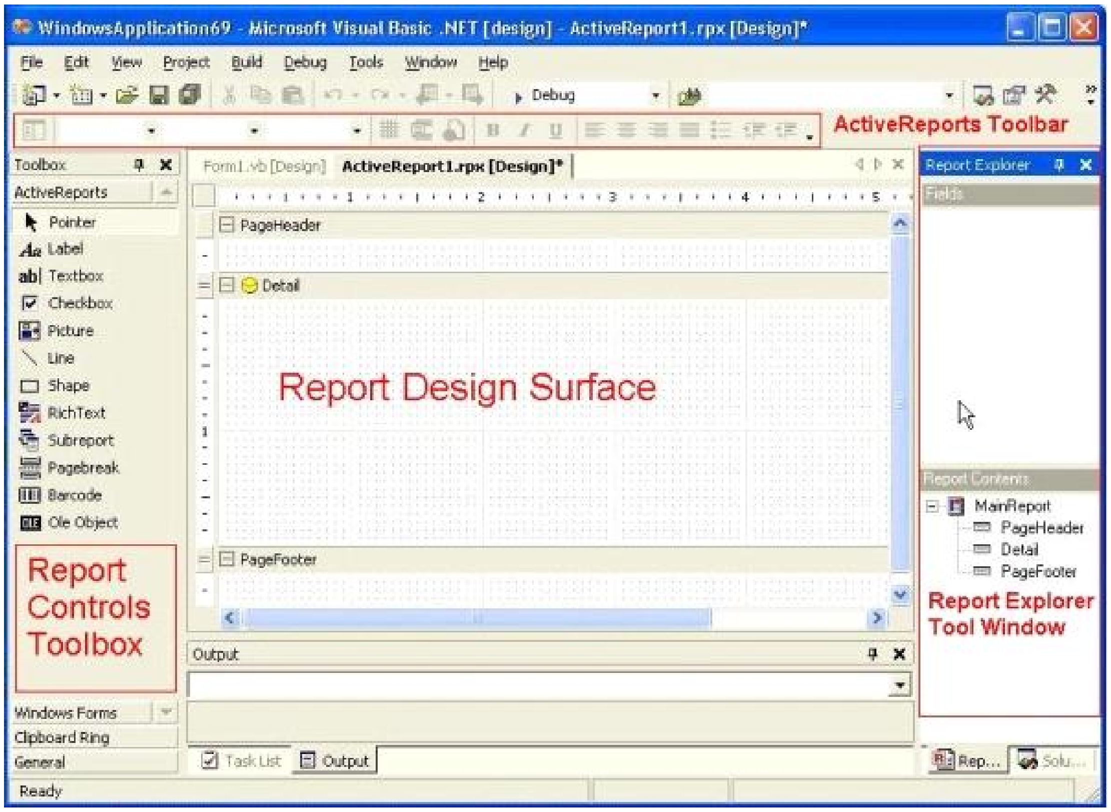


Create a new ActiveReports File and open it in the report designer

Name: ActiveReport1.rpx

4. Click **Open** to add the report to your project.

Visual Studio creates and displays the following ActiveReports designer document.



Binding Reports to a Data Source

ActiveReports allows much flexibility in binding reports to various kinds of data sources. In this section, you will learn how to use various methods to bind reports to data sources.

Note DAO and RDO data controls are no longer supported in ActiveReports for .NET. The ADO data control is converted to an ADO.NET data source. Data controls are removed and replaced with a report data source dialog accessible from the data source icon on the detail section of the report. The XML Data control is converted into an XML data source.

- Using a Data Set
- Using a Data View
- Using the Data Source Icon

Data Set

In addition to being able to set the reports data source at design time, it is also possible to set the reports data source at run time to a data set. This makes it easy to use data sets created with Microsofts data controls in your reports. To use a data set, set the reports DataSource property to the data set being used and the reports DataMember property to the table from which the reports data is derived.

The following example shows what the code for the method looks like:

```
[Visual Basic]
```

```
        Dim rpt As New rptDataView()
        rpt.DataSource = Me.dataSet11
        rpt.DataMember = "employees"

[C#]

         rptDataView rpt = new rptDataView();
         rpt.DataSource = this.dataSet11;
         rpt.DataMember = "employees";
```

Data View

In addition to using a data set, the reports data source can be set to a data view. This can be useful for creating reports containing filtered information. To use the data view in the report, set the reports DataSource property to the data view created from the filtered data set (see Using a Data Set for more information).

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_dbPath As String
Dim usView As New DataView()
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As _
         System.EventArgs)Handles MyBase.Load
        m_dbPath = getDatabasePath()
        Me.oleDbConnection1.ConnectionString = _
         Data Source=" + m_dbPath + "\\NWIND.MDB;Persist                                      Security
        Me.oleDbDataAdapter1.Fill(Me.dataSet11)
        usView = me.dataSet11.Tables("employees"))
        usView.RowFilter = "Country = 'USA'"
        Me.dataGrid2.DataSource = usView
End Sub

[C#]

DataView usView;
private void Form1_Load(object sender, System.EventArgs e)
{
         string m_dbPath = getDatabasePath();
         this.oleDbConnection1.ConnectionString =
                  Data Source=" + m_dbPath + "\\NWIND.MDB;Persist
         this.oleDbDataAdapter1.Fill(this.dataSet11);
         usView = new DataView(this.dataSet11.Tables["employees"]);
         usView.RowFilter = "Country ='USA'";
         this.dataGrid2.DataSource = usView;
}
```

DataSource Icon

ActiveReports makes it easy to bind your report to a data source by using the yellow DataSource icon located in the Detail section of the report design surface or by accessing the DataSource dialog from the Report Settings menu.

To use the DataSource icon

1. Open a Visual Studio project.
2. Add an ActiveReport to your project.
3. Once the report is added, you will see the report design surface.

4. Click on the yellow DataSource icon in the Detail section of the report.
5. You will then be prompted to select your data source, connection string and query.

To connect to Microsoft Access using Jet 4.0

1. Click on the yellow report DataSource icon in the Detail section.
2. This brings up the report DataSource dialog box.
3. Click on **Build...**
4. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
5. Enter a database name or click on the ellipsis to browse for the access path to a database.
6. Click **Open** once you have entered a database name or selected the appropriate access path.
7. Click **OK** to continue.
8. Enter a SQL statement in the Query box (e.g. "Select * from products").
9. Click **OK** to return to the report design surface.

To connect to SQL Server

1. Click on the yellow report DataSource icon in the Detail section.
2. This brings up the report DataSource dialog box.
3. Click on the Data Source drop-down arrow and select *SqlClient*.
4. Click on **Build...**
5. Select "Microsoft OLE DB Provider for SQL Server" and click **Next >>**.
6. Select a server.
7. Chose Windows integrated security or a specific username and password.
8. Choose the database for connection and click **OK**.
9. Enter a SQL statement in the Query box (e.g. "Select * from products").
10. Click **OK** to return to the report design surface.

To connect to a XML database

1. Click on the yellow report DataSource icon in the Detail section.
2. This brings up the report DataSource dialog box.
3. Click on the Data Source drop-down arrow and select *XML*.
4. Click on the ellipsis beside *File URL* to browse for the access path to Customer.xml.
5. Click **Open** once you have selected the appropriate access path.
6. In the *Recordset Pattern* field, enter a pattern (e.g. "//ITEM").
7. Click **OK** to return to the report design surface.

Grouping Data

In ActiveReports, a report can consist of single or multiple nested groups, with each group having its own header and footer sections. The header section is inserted and printed immediately before the Detail section. The footer section is inserted and printed immediately after the Detail section. Up to 32 nested groups are allowed in a single report.

> **Note** ActiveReports does not order records for grouping. It assumes the data is already sorted in the same grouping order. The data source needs to be ordered by the field on which you want your data grouped to achieve the desired results.

Grouping Data in a Report

To group data in a report

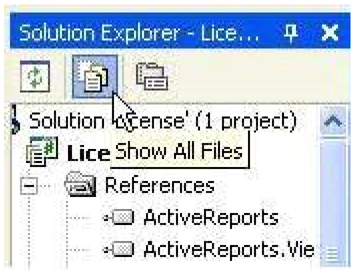
1. Right-click in the Detail section of the report design surface, select **Insert**, and click on **Group Header/Footer**.
2. This will insert a new group header/footer section into your report.
3. In the Properties window for the group header, change the DataField property to the field on which you want your data grouped.
4. Change the name of the group header to reflect the field on which data is being grouped. For example, "ghCategories" would be the name of the group header field with the DataField property of "CategoryID".

Licensing Applications

Checking ActiveReports Windows Applications for Licensing

To check an existing ActiveReports Windows application for licensing

1. Open an existing ActiveReports Windows application project.
2. In the Solution Explorer window, choose the "Show All Files" icon.



3. If the ActiveReports application is licensed, you will see a file called "licenses.licx".
4. If the "licenses.licx" file is not listed in your application's file list, you will need to manually set up the application for licensing.

Manually Licensing Windows Applications

To manually set up Windows applications for licensing

1. Open an existing ActiveReports project or create a new one.
2. On the Project menu, select Add New Item...
3. In the Templates window, choose **Text File**.
4. Change the name of the text file to "licenses.licx".
5. This adds the "licenses.licx" file to Solution Explorer. Double-click "licenses.licx" to open the file.
6. Add the following line to the text file: "DataDynamics.ActiveReports.ActiveReport, ActiveReports".
7. Save your project. Your ActiveReports Windows application will now be licensed.

Licensing Web Applications

To set up Web applications for licensing

1. Open an existing ActiveReports Web application.
2. From the **Start** Menu, click **All Programs** > **Data Dynamics** > **ActiveReports.NET** > "Create Web.Config Key".
3. In the **Data Dynamics ActiveReports Web Key Generator** dialog, enter your name, company and the serial number.
4. Click on "Create Web.Config".
5. Copy the contents of the generated Web key text.
6. In your ActiveReports Web application, double-click the Web.config file to open it.
7. In the XML view of the Web.config file, paste the contents of the generated Web key text between <configuration> and <system.web>.
8. Save the project. Your ActiveReports Web application will now be licensed.

Localizing the Viewer Control

In ActiveReports, you can localize settings for the Windows Forms Viewer control by modifying a provided "strings" text file, generating a resources file, embedding the resources file in your ActiveReports project and adding the localization code needed in your Form_Load event.

Making localization changes to the strings text file

To make changes to the text file

1. Browse to the RDF Viewer sample in the <Installation folder>\Samples\VB folder.
2. Double-click the “Strings.txt” file to open the file in Notepad.
3. Make changes to localize settings for the viewer control.
4. Click on **Save As...**, rename your text file "localization.txt" and save it to your location of choice.

Generating the resources file from the text file

To generate the resources file

1. From the Start bar, click on **All Programs > Microsoft Visual Studio.NET > Visual Studio .NET Tools > Visual Studio .NET Command Prompt**.
2. Change the prompt path to reflect the location of your localization.txt file.
3. Type "resgen localization.txt localization.resources" **.**
4. This creates a resources file in the same location as the text file.

Adding the resources file to your Windows application

To add the resources file

1. Open your ActiveReports Windows application which includes a Windows Form with an ActiveReports Windows Forms Viewer control.
2. In the Solution Explorer window, click on the icon to "Show All Files".
3. Right-click on the name of your project and click **Add** > **Add Existing Item...**
4. Select the appropriate file path to the localization.resources file and click **Open**.
5. This adds the resources file to your application.

Adding code to the Form1_Load event

To write the code for the viewer in Visual Basic

- Right-click on Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event.

To write the code for the viewer in C#

- Click on the blue section at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for Form1. Double-click *Load*. This creates an event-handling method for the Form1_Load event.

The following example shows what the code for the method looks like:

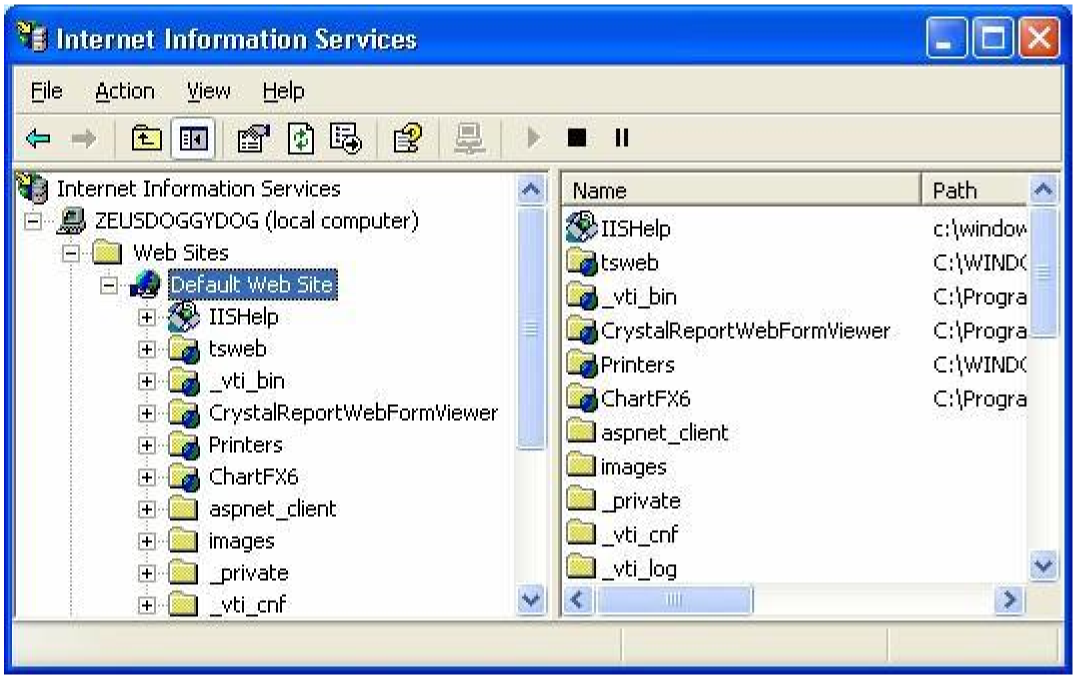
```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As _ System.EventArgs)
Handles MyBase.Load
          Dim res As New ResourceManager("rptLocalize.localization",
        Me.Viewer1.Localize = res
        Dim rpt As New rptLocalize()
        Viewer1.Document = rpt.Document
        rpt.Run(True)
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         ResourceManager res = new
         this.viewer1.Localize = res;
         rptLocalize rpt = new rptLocalize();
         viewer1.Document = rpt.Document;
         rpt.Run(true);
}
```

Manually Configuring Web Samples

Installing the Sample Web Application

To install the sample web application

1. In the Control Panel, double-click "Administrative Tools." Double-click "Internet Information Services" to open its dialog window.



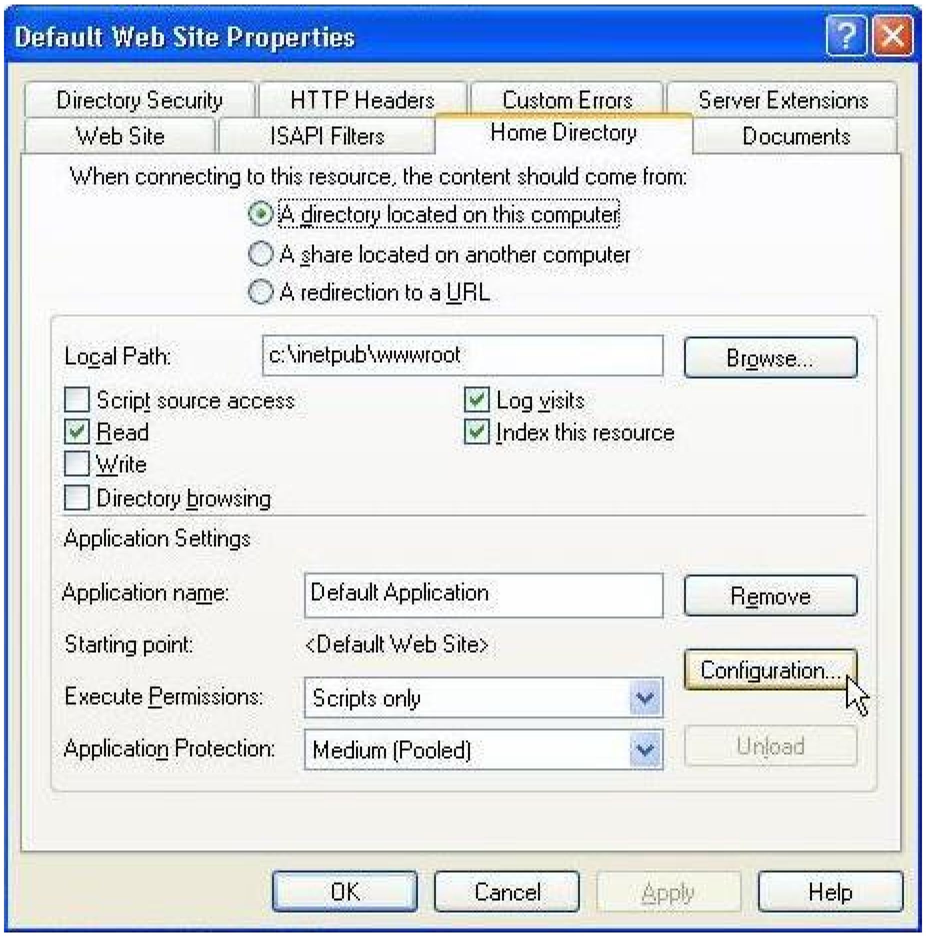
2. Right-click on "Default Web Site" then click **New > Virtual Directory...**
3. Click **Next** when you see the "Virtual Directory Creation Wizard."
4. In the Alias field, enter one of the following folder names, depending on which web sample you need to configure:
   - ArWebSampleStdCs
   - ArWebSampleStdVb
   - ArWebSampleProCs
   - ArWebSampleProVb
5. Click **Next** to continue.
6. Click **Browse...** to find the folder in which ActiveReports is installed. Find and select the appropriate folder.
7. Click **OK** to continue and then click **Next**.
8. Make sure the "Read" and "Run Scripts" permissions are checked and click **Next**.
9. The ActiveReports Web sample is now installed.

Note If you are only configuring Standard Edition Web Samples, you do not need to complete the following steps.

Configuring the ActiveReports HTTPHandlers

To configure the HTTPHandler

1. In the Control Panel, double-click "Administrative Tools." Double-click "Internet Information Services" to open its dialog window.
2. Right-click on "Default Web Site" then click **Properties**.
3. Click on the "Home Directory" tab of the "Default Web Site Properties" dialog.
4. Click on the Configuration button.


Default Web Site Properties

When connecting to this resource, the content should come from:
- [✓ A directory located on this computer]
- A share located on another computer
- A redirection to a URL

Local Path: c:\inetpub\wwwroot

- ☐ Script source access
- ✓ Read
- ☐ Write
- ☐ Directory browsing
- ✓ Log visits
- ✓ Index this resource

Application Settings

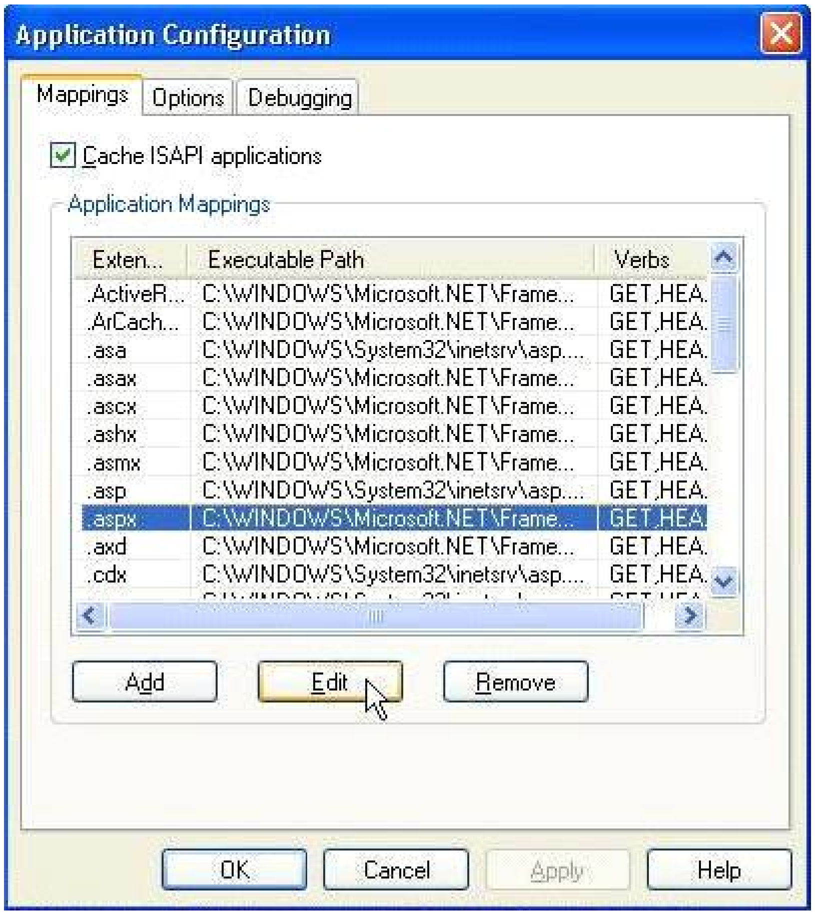
Application name: Default Application

Starting point: <Default Web Site>

Execute Permissions: Scripts only

Application Protection: Medium (Pooled)

5. In the "Application Mappings" window of the Application Configuration dialog, select the list item with .aspx in the extension column and click **Edit**.

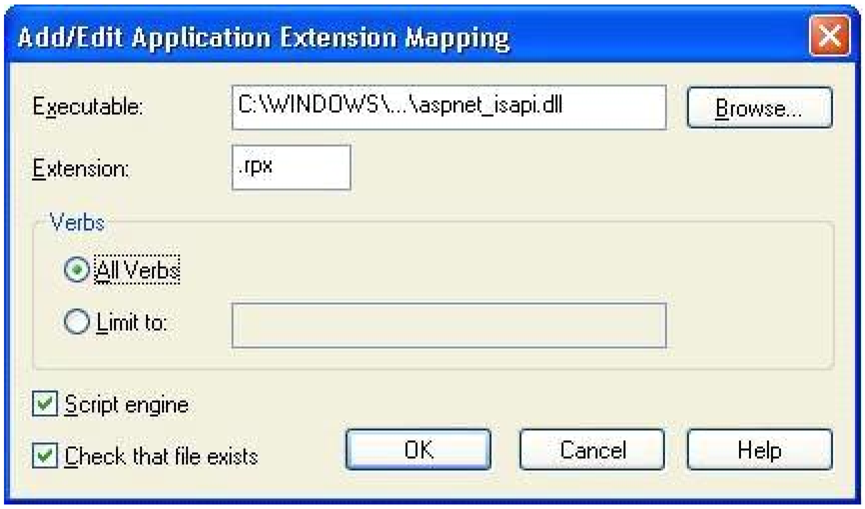


☑ Cache ISAPI applications

Application Mappings

Exten...	Executable Path	Verbs
.ActiveR...	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.ArCach...	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.asa	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA.
.asax	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.ascx	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.ashx	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.asmx	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.asp	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA.
.aspx	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.axd	C:\WINDOWS\Microsoft.NET\Frame...	GET,HEA.
.cdx	C:\WINDOWS\System32\inetsrv\asp....	GET,HEA.

6. Select and copy all of the text in the "Executable" field. Click **Cancel** to return to the Application Configuration dialog.
7. Click **Add** in the Application Configuration dialog to add a new Application Mapping.
8. In the Executable field, paste the value copied from Step 6 and enter ".rpx"in the Extension field.



Executable: C:\WINDOWS\...\aspnet_isapi.dll

Extension: .rpx

Verbs

● All Verbs

○ Limit to:

☑ Script engine

☑ Check that file exists

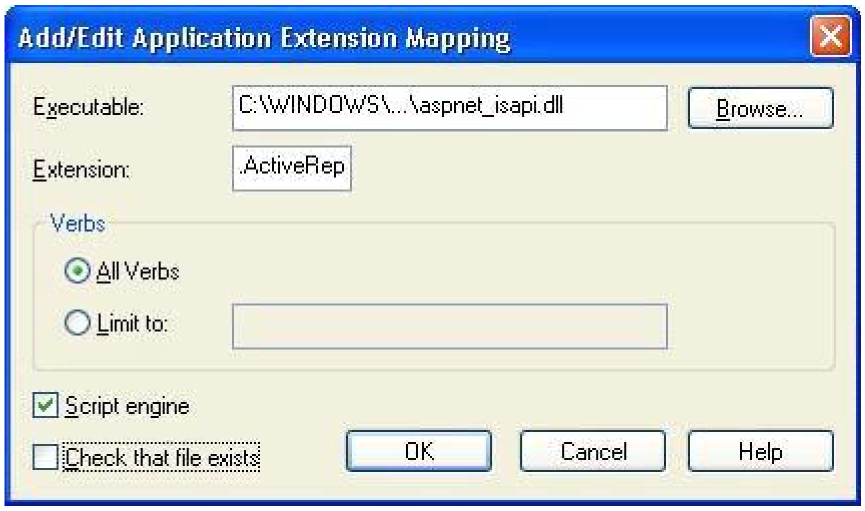
9. Click **OK** to add the mapping and return to the Application Configuration dialog.

To configure the compiled report handler (continuing from Step 9 in "To Configure the HTTPHandler)

1. In the Application Configuration dialog, click **Add** to add a new Application Mapping.
2. In the Executable field, paste the value copied from Step 6 above.
3. Enter ".ActiveReport" in the Extension field.
4. Make sure the "Check that file exists" permission is unchecked.
5. Click **OK** to add the mapping and return to the Application Configuration dialog.

To configure the WebCacheAccessHandler (continuing from Step 9 in "To Configure the HTTPHandler)

1. In the Application Configuration dialog, click **Add** to add a new Application Mapping.
2. In the Executable field, paste the value copied from Step 6 above.
3. Enter ".ArCacheItem" in the Extension field.
4. Make sure the "Check that file exists" permission is unchecked.



Executable: C:\WINDOWS\...\aspnet_isapi.dll
Extension: .ActiveRep
All Verbs [✓]
Limit to:
Script engine [✓]
Check that file exists [ ]

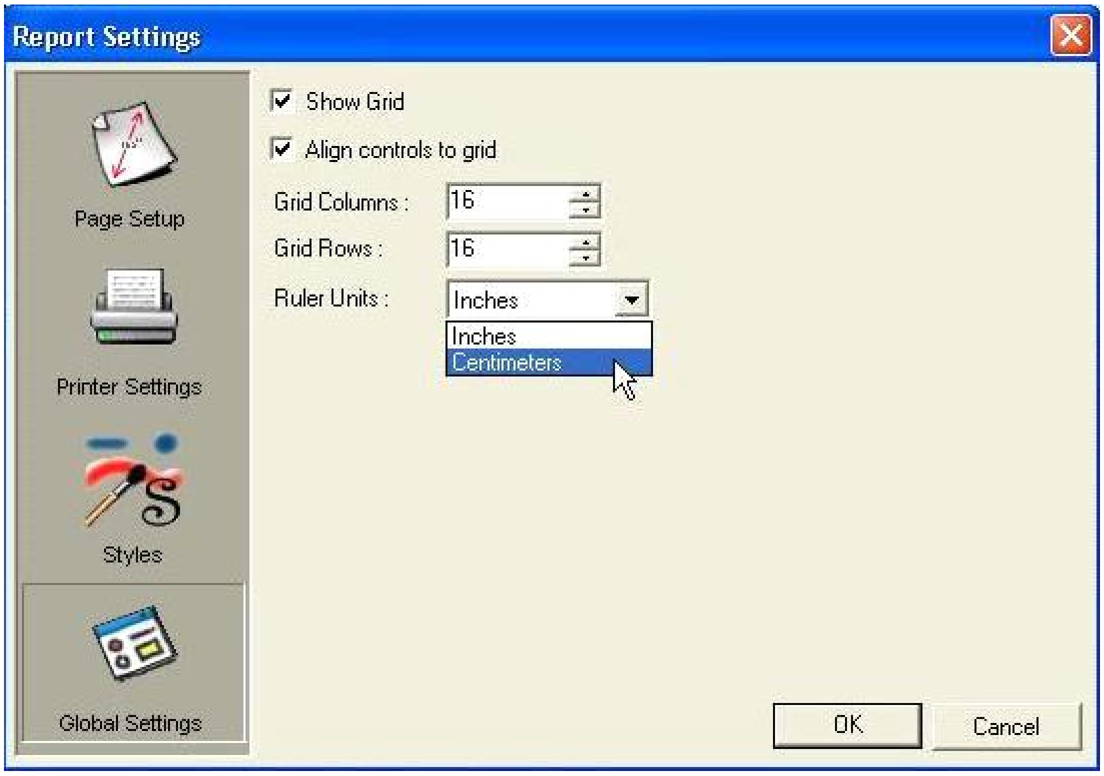
5. Click **OK** to add the mapping and return to the Application Configuration dialog.
6. Click **OK** on the remaining open dialogs to exit the IIS Administrative tool.

Metric Units

In ActiveReports, ruler measurements can be changed from inches to centimeters and centimeters to inches from design time. Conversion values for centimeters to inches or inches to centimeters can be called at run time as well.

To change ruler measurements at design time

1. In an existing ActiveReports project, click on **Report**, **Settings...**
2. In the Report Settings dialog, click on Global Settings.
3. Change Ruler Units from inches to centimeters or centimeters to inches.



☑ Show Grid

☑ Align controls to grid

Grid Columns : 16

Grid Rows : 16

Ruler Units : Inches

Inches
Centimeters

To call a measurement conversion at run time

Call the CmToInch method or InchToCm method whenever needed. For example, if you were working in centimeters and needed to convert a label's position measurements from centimeters to inches at run time, you would use the following code.

```
[Visual Basic]

Me.lblMyLabel.Left = ActiveReport.CmToInch(2)
Me.lblMyLabel.Top = ActiveReport.CmToInch(2)

[C#]

this.lblMyLabel.Left = ActiveReport.CmToInch(2);
this.lblMyLabel.Top = ActiveReport.CmToInch(2);
```

Saving and Loading RDF Files

ActiveReports allows reports to be saved into their own standard format called an RDF file (Report Document Format). Once a report has been saved to an RDF file, it can be loaded into the viewer control and used to display reports in custom preview applications.

To write the code to save a report as an RDF file in Visual Basic

- Right-click in any section of the Windows Form, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the saveRDF event.

To write the code to save a report as an RDF file in C#

- Double-click on the Windows Form to see the code view for the Windows form. Add the following code to create the saveRDF event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim rpt As New ActiveReport1()
Private Sub saveRDF()
        rpt.Run()
        rpt.Document.Save(Application.StartupPath + "\\NewRDF.RDF")
End Sub

[C#]

private void saveRDF()
{
         ActiveReport1 rpt = new ActiveReport1();
         rpt.Run();
         rpt.Document.Save(Application.StartupPath + "\\NewRDF.RDF");
}
```

To write the code to load the saved RDF into the ActiveReports viewer in Visual Basic

- Right-click in any section of the Windows Form, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the loadRDF event.

To write the code to load the saved RDF into the ActiveReports viewer in C#

- Double-click on the Windows Form to see the code view for the Windows form. Add the following code to create the loadRDF event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub loadRDF()
        rpt.Run()
        Viewer1.Document.Load(Application.StartupPath +                                        "\\NewRDF.
End Sub

[C#]

private void loadRDF()
{
         ActiveReport1 rpt = new ActiveReport1();
         rpt.Run();
         viewer1.Document.Load(Application.StartupPath +
}
```

Saving and Loading RPX Files

To write the code to save a report as an RPX file in Visual Basic

- Right-click in any section of the Windows Form, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the saveRPX event.

To write the code to save a report as an RPX file in C#

- Double-click on the Windows Form to see the code view for the Windows form. Add the following code to create the saveRPX event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim rpt As New ActiveReport1()
Private Sub saveRPX()
        rpt.Run()
        rpt.SaveLayout(Application.StartupPath + "\\NewRPX.RPX")
End Sub

[C#]

private void saveRPX()
{
         ActiveReport1 rpt = new ActiveReport1();
         rpt.Run();
         rpt.SaveLayout(Application.StartupPath + "\\NewRPX.RPX");
}
```

To write the code to load the saved RPX into the ActiveReports viewer in Visual Basic

- Right-click in any section of the Windows Form, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the loadRPX event.

To write the code to load the saved RPX into the ActiveReports viewer in C#

- Double-click on the Windows Form to see the code view for the Windows form. Add the following code to create the loadRPX event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub loadRPX()
        Viewer1.Document = rpt.Document
        rpt.Run()
End Sub

[C#]

private void loadRPX()
{
         ActiveReport1 rpt = new ActiveReport1();
         viewer1.Document = rpt.Document;
         rpt.Run();

}
```

Note When saving to an RPX file, make sure you save the report before it runs. Saving the layout after the report runs will result in unwanted data being saved in the layout. If calling saveRPX inside the report, use the ReportStart event. Also, when saving the report layout, the script code is the only code that is saved to the file. The code in the reports .cs or .vb file will not be saved to the RPX file.

Getting Assistance

This topic will show you how to locate Readme information as well as how to obtain support for ActiveReports for .NET.

- Readme File
- Product Support

Readme File

The file, Readme.html, will be located in the “Introduction” folder in the root ActiveReports for .NET installation folder on your computer once the program is installed. Use your Internet browser to open and view the file.

Product Support

This topic will explain how to register your ActiveReports for .NET purchase and obtain technical support.

In this section

- Product Registration
- Technical Support

Product Registration

In order to receive telephone support, product news and upgrade announcements, you must register your product purchase with Data Dynamics. We encourage you to register your purchase as soon as you receive it using any of the following:

- Fill out the enclosed registration card and mail it to Data Dynamics, 5870 Cleveland Avenue, Columbus, Ohio 43231
- Fax the registration card to Data Dynamics at (614) 899-2943
- Complete the registration form on our website at http://www.datadynamics.com/register/default.htm

Technical Support

Technical support is available for ActiveReports for .NET in a variety of media.

Telephone Support

Telephone support is available for registered users of ActiveReports for .NET for up to five support incidents. Additional support requests should be directed to the appropriate newsgroup. If desired, additional telephone support can be acquired by purchasing any of the support packages available through Data Dynamics. Contact sales@datadynamics.com for details.

When contacting Data Dynamics with support questions, be prepared to provide a serial number, the full version number of ActiveReports, a complete description of the problem and hardware and operating environment specifications.

Support telephone number: (614) 895-3142 (9:00am-5:00pm EST)

E-mail Support

E-mail support is available for ActiveReports for .NET. Contact activereports.support@datadynamics.com.

Website

The Data Dynamics website offers the latest product news, white papers, tutorials, report samples and product service packs. Please visit the website for the latest news about ActiveReports for .NET before contacting technical support.

Product Upgrades

Minor upgrades and service packs will be made available for download from the Data Dynamics website free of charge.

http://www.datadynamics.com/downloads.asp

Major upgrades will carry an upgrade price that is determined separately for each release. You will be eligible for a free major upgrade if you purchased the product within 30 days of the upgrade release date.

KnowledgeBase articles

The Data Dynamics KnowledgeBase contains hundreds of helpful articles for all Data Dynamics products. You can search the entire KnowledgeBase for keywords or narrow down your search first by choosing a specific product before submitting your search criteria.

http://www.datadynamics.com/kb

Newsgroups

The Data Dynamics news server can be used to read and post questions and answers about issues you encounter with ActiveReports for .NET. Tips and tricks can be communicated with other users and access to the Data Dynamics technical support team can be gained in an online community forum. Data Dynamics' technical support engineers monitor the newsgroups constantly and are available to answer questions and assist with any issues encountered using the product.

Product announcements will also be posted to the news server.

Newsgroup address for ActiveReports for .NET:
news://news.datadynamics.com/support.activereports.NET

Samples

Standard Edition Samples

In this section

- **Annual Report**--demonstrates subreports, nested subreports, modifying data source properties at run time, alternate row highlighting and pagebreak control.
- **ASP.NET Standard Edition Web Sample**--demonstrates using Standard Edition in ASP.NET. It shows how to use custom exporting without the Pro Edition server controls or RPX handlers as well as running reports on the server, exporting output to HTML or PDF streams and pushing content to the client. The sample also demonstrates using the ActiveX viewer control to view report output on the client machine.
- **Category Selection**--demonstrates using the ad hoc report filter by modifying the report's SQL query at run time.
- **Cross-Tab**--demonstrates using unbound data, conditional highlighting and distributing data across columns to create a cross-tab view and data aggregation.
- **Custom Preview**--demonstrates using viewer control customization, export filters, rich edit control and mail-merge, and grouping.
- **Data Reader Binding**--demonstrates binding to an ADO.NET DataReader object.
- **Data View Binding**--demonstrates binding to an ADO.NET DataView object.
- **DataGrid Printing**--demonstrates creating an ActiveReport object dynamically to print a DataGrid control.
- **E-mail**--demonstrates rendering a stand-alone RPX file to PDF and e-mailing the PDF file as an attachment using the default mail client.
- **Hyperlinks and Drill-Downs**--demonstrates using hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another.
- **RDF File Viewer**--demonstrates customizing the WinForms viewer control toolbar, loading Report Document Files (RDF) and using the export filters.
- **Report Assemblies**--demonstrates distributing reports as separate assembly files and calling them from the main application .exe file.
- **Unbound From Array**--demonstrates retrieving data from an array in unbound mode.
- **Unbound From Text File**--demonstrates retrieving data from a text file in unbound mode.
- **XML Data**--demonstrates the XML data source and using it to run multi-level reports with and without using subreports.

Professional Edition Samples

In this section

- **ASP.NET Web Sample**--demonstrates the use of Professional Edition ASP.NET features, including RPX HTTP Handlers, Report Caching and the Server Viewer Control.
- **End-User Report Designer Control**--demonstrates a custom end-user report designer that can be integrated in your applications to allow users to modify report layouts.

Walkthroughs

Standard Edition Walkthroughs

Walkthroughs give basic step-by-step instructions for common situations in ActiveReports. This feature makes walkthroughs an excellent way to learn more about the basic features of the product.

In this section

- **Advanced Report Layouts**--describes how to create advanced report layouts.
- **Bookmarks**--describes how to set up Bookmarks to organize reports for easy navigation.
- **Calculated Fields**--describes how to use a field's DataField property to perform calculations in a report.
- **Conditional Formatting**--describes how to use the format event to modify the properties of sections or controls at run time.
- **Custom Exporting**--describes how to set up custom exporting to HTML and PDF on the web.
- **Customizing the Viewer Control**--describes how to customize the viewer control in a report.
- **Data Bound Reports**--describes how to connect a report to a data source using the DataSource icon.
- **Deploying Compiled Reports**--describes how to deploy compiled reports.
- **Exporting Output**--describes how to use export filters to make reports available in other formats.
- **Grouping Data**--describes how to incorporate grouping in a report.
- **Hyperlinks**--describes the incorporation and function of hyperlinks in a report.
- **Master Detail Reports**--describes how to implement master detail reports.
- **Modifying Report Documents**--describes several methods of modifying report documents.
- **Page Numbering**--describes how to use page numbering in the page footer and in the group header.
- **Parameters**--describes how to use parameters with simple reports and subreports.
- **Printing**--describes how to use different printing functions in a report.
- **Rich Text and Field Merging**--describes how to use Rich Text and field merging in a report.
- **Run-Time Reporting**--describes how to create dynamic reports at run time.
- **Saving and Loading to a Memory Stream**--describes how to save reports to and load reports from a memory stream.
- **Scripting**--describes how to use scripting to generate stand-alone reports, including subreports.
- **Style Sheets**--describes how to use style sheets in a report.
- **Subreports**--describes how to add subreports to a report.
- **Summary Fields**--describes how to add summary fields to a report to calculate totals, counts, averages and other aggregations.
- **Unbound Reports**--describes the basics of using the Data_Initialize and Fetch_Data events to connect a report to a data source.

ActiveX Viewer Control on the Web

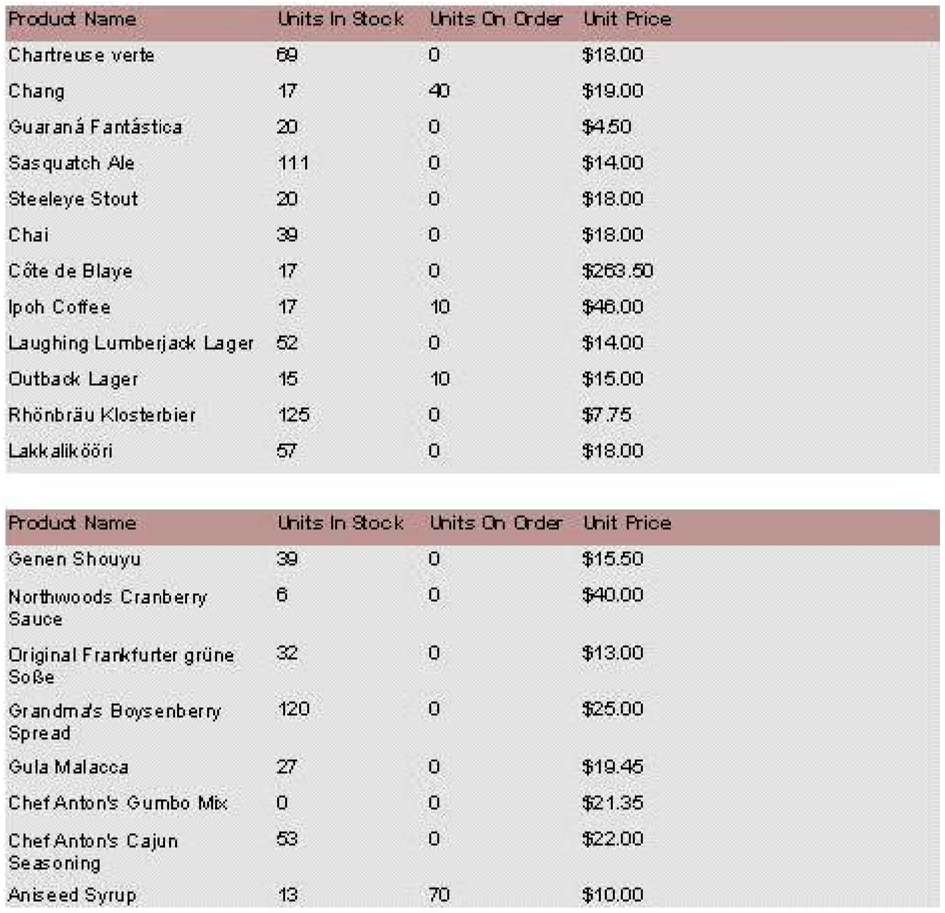
The ActiveX Viewer Control allows you to view and print report output in a web browser.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to an ASP.NET Web application
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding the ActiveX viewer .cab file to the project folder
- Adding a ReportOutput folder to the project folder
- Adding required Object tags to the html code for the Web Form
- Adding code for the window_onload event for the Web Form
- Adding code to the Web Form's Page_Load event

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb). You must also have access to Internet Information Services either from your computer or from the server. You must also run the "Configure Web Sample" option from the Data Dynamics ActiveReports for .NET program menu from your Windows Start button.

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product Name	Units In Stock	Units On Order	Unit Price
Chartreuse verte	69	0	$18.00
Chang	17	40	$19.00
Guaraná Fantástica	20	0	$4.50
Sasquatch Ale	111	0	$14.00
Steeleye Stout	20	0	$18.00
Chai	39	0	$18.00
Côte de Blaye	17	0	$263.50
Ipoh Coffee	17	10	$46.00
Laughing Lumberjack Lager	52	0	$14.00
Outback Lager	15	10	$15.00
Rhönbräu Klosterbier	125	0	$7.75
Lakkaliköőri	57	0	$18.00

Product Name	Units In Stock	Units On Order	Unit Price
Genen Shouyu	39	0	$15.50
Northwoods Cranberry Sauce	6	0	$40.00
Original Frankfurter grüne Soße	32	0	$13.00
Grandma's Boysenberry Spread	120	0	$25.00
Gula Malacca	27	0	$19.45
Chef Anton's Gumbo Mix	0	0	$21.35
Chef Anton's Cajun Seasoning	53	0	$22.00
Aniseed Syrup	13	70	$10.00

Adding an ActiveReport to an ASP.NET Web application

To add an ActiveReport to your project

1. Open a new ASP.NET Web application in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptActiveX.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**.
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products order by categoryID".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptActiveX.
2. Make the following changes to the group header:
   - Change the name to ghCategories
   - Change the DataField property to CategoryID
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductName**	Product Name	0, 0
Label	**lblUnitsInStock**	Units In Stock	1.875, 0
Label	**lblUnitsOnOrder**	Units On Order	2.9375, 0
Label	**lblUnitPrice**	Unit Price	4, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	1.875, 0	(Empty string)
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	2.9375, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4, 0	Currency

Adding the ActiveX .cab file to the project folder

To add the ActiveX .cab file

1. Open Windows Explorer and browse to the folder in which ActiveReports for .NET is installed.
2. Double-click the Deployment folder. Copy the file called "arview2.cab" by right-clicking on the file and selecting **Copy**.
3. Browse to the folder in which your project is contained.
4. Paste the .cab file into your project's folder.

Adding a ReportOutput folder to the project folder

To add a folder to the project

1. Open Windows Explorer and browse to the folder in which your project is contained.
2. On the **File** menu, click **New**, **Folder**.
3. Name the folder "ReportOutput".
4. Make sure that you have write permissions for this folder.

Adding Object tags to the Web Form's HTML code

To add Object tags to the Web Form

In the body of the HTML view of the Web Form, add the following code:

<OBJECT id="arv" codeBase="arview2.cab" height="100%" width="100%"
classid="clsid:8569D715-FF88-44BA-8D1D-AD3E59543DDE" VIEWASTEXT>

```
          <PARAM NAME="_ExtentX" VALUE="11218">
          <PARAM NAME="_ExtentY" VALUE="7329">
</OBJECT>
```

Adding code to the Web Form's window_onload event

To add code to the window_onload event

1. At the top of the HTML view of the Web Form, click on the drop-down arrow for "Client Objects and Events" and select "window".
2. Click the drop-down arrow for the available events to the right of "window" and select "onload."
3. This creates an event-handling method for the Web Form's window_onload event.
4. Add the following code to the window_onload event:

```
          arv.datapath = "ReportOutput/axreport.rdf";
```

Adding code to the Web Form's Page_Load event

To write the code in Visual Basic

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

To write the code in C#

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Page_Load(ByVal sender As System.Object, ByVal  _
          e As System.EventArgs)Handles MyBase.Load
        Dim rpt As New rptActiveX()
        rpt.Run()
        rpt.Document.Save(Server.MapPath("") +                                                 "\ReportOu
End Sub

[C#]

private void Page_Load(object sender, System.EventArgs e)
{
          rptActiveX rpt = new rptActiveX();
          rpt.Run();
          rpt.Document.Save(Server.MapPath("") +
}
```

Advanced Report Layouts

ActiveReports provides flexibility in creating advanced report layouts. The following information demonstrates how to set up different types of reports based on the output needed.

Column Reports

ActiveReports supports newspaper column layout in both the Detail and Group sections. You can render the columns either horizontally or vertically in the section with options to break the column on Group section (i.e. start a new column on the change of a group).

Labels

ActiveReports can be used to print any label size by using the newspaper column layout.

- To create a report that prints labels to a laser printer labels sheet:
- Set the report width to the total width of the label sheet.
- Set the columns property of the Detail section to the number of labels across the page.
- Adjust the column spacing property, if needed, to increase/decrease the space between columns.
- Remove the page header and footer sections from your report.
- Set the height of the Detail section to the exact height of the label.
- Set the Detail section's CanGrow/CanShrink properties to False.
- Adjust the top, right, bottom and left margins to match the sheet.
- Finally, place your text and label controls in the column area of the Detail section.

ActiveReports allows you to skip labels on a label sheet using the LayoutAction property.

LayoutAction can be used to skip a section (i.e. do not print anything and move to the next printable area). For each label you wish to skip, set the LayoutAction property to MoveLayout in the Format event of the Detail section.

The section will be skipped without moving to the next record in the data source. See the following code:

```
[Visual Basic]

Dim iSkipLabels As Integer
Private Sub Detail_Format(ByVal sender As Object, ByVal e _
         As System.EventArgs)Handles Detail.Format
         If iSkipLabels > 0 Then
                  iSkipLabels = iSkipLabels - 1
                  LayoutAction = LayoutAction.MoveLayout
         End If
End Sub

[C#]

int iSkipLabels;
private void Detail_Format(object sender, System.EventArgs eArgs)
{
         If (iSkipLabels > 0);
         {
                  iSkipLabels = iSkipLabels - 1;
                  LayoutAction = MoveLayout;
         }
}
```

Top N Reports

ActiveReports requires no special handling for Top N Records report styles. Such reports can be easily implemented by setting the data source to a Top N filtered query. If the data source does not support Top N queries, the query can be set to return records ordered by the Top N value descending. Then the MaxRows property should be set to N.

For example, to list the top 10 customers by their sales numbers, you can create a query that returns all customer sales ordered by the sales value descending. Then set the MaxRows property of the data control to 10. ActiveReports will process only 10 records from the sorted query results.

Summary Reports

Summary reports are implemented by setting the Visible property for the Detail section to False or setting the Height to 0. The Detail section will be processed and the summary GroupHeader and Footer sections will be printed without the Detail section.

Green-Bar Reports

Green-bar printouts can be created by alternating the shading or background color of the report's Detail section in the Format event.

See the following code:

```
[Visual Basic]

Dim I As Integer
Private Sub Detail_Format(ByVal sender As Object, ByVal e _
         As System.EventArgs)Handles Detail.Format
         If (I Mod 2) = 0 Then
                  Detail.BackColor = System.Drawing.Color.DarkSeaGreen
         Else
                  Detail.BackColor = System.Drawing.Color.Transparent
         End If
         I = I + 1
End Sub
```

```
[C#]

bool m_color;
private void Detail_Format(object sender, System.EventArgs eArgs)
{
         if(m_color)
         {
                  m_color =false;
                  this.Detail.BackColor =
         }
         else
         {
                  this.Detail.BackColor =
                  m_color = true;
         }
}
```

Charts

ActiveReports does not include a built-in chart control. However, it allows you to use any charting control in your report. You can use the data in your report to set series and data points in the chart as the report is being processed.

If you place the chart in the report header (i.e. before the data is processed), you will need to place a summary field control in the same section. This allows ActiveReports to delay printing the section until all the required data is processed and you will get a chance to load your chart data correctly.

Another alternative is to place the chart control in a child report and link it to a Subreport control in the main report. This allows you to fully process the data for the chart and then render it onto the main report using the Subreport control. However, this would require going through more than one set of records, one for the main report and another for the child report.

Bookmarks Walkthroughs

Setting up Bookmarks (formerly called Table of Contents) allows reports to be organized and easily navigated. By default, no Bookmarks are created when a report is run. However, by adding simple code to the desired section event, Bookmark entries can be set up as the report runs.

- Bookmarks with Grouping
- Bookmarks with Simple Reports
- Bookmarks with Subreports

Bookmarks with Grouping

ActiveReports allows Bookmarks to be easily set up and used with grouping by adding code to the Detail_Format event of the report and in the group header Format event.

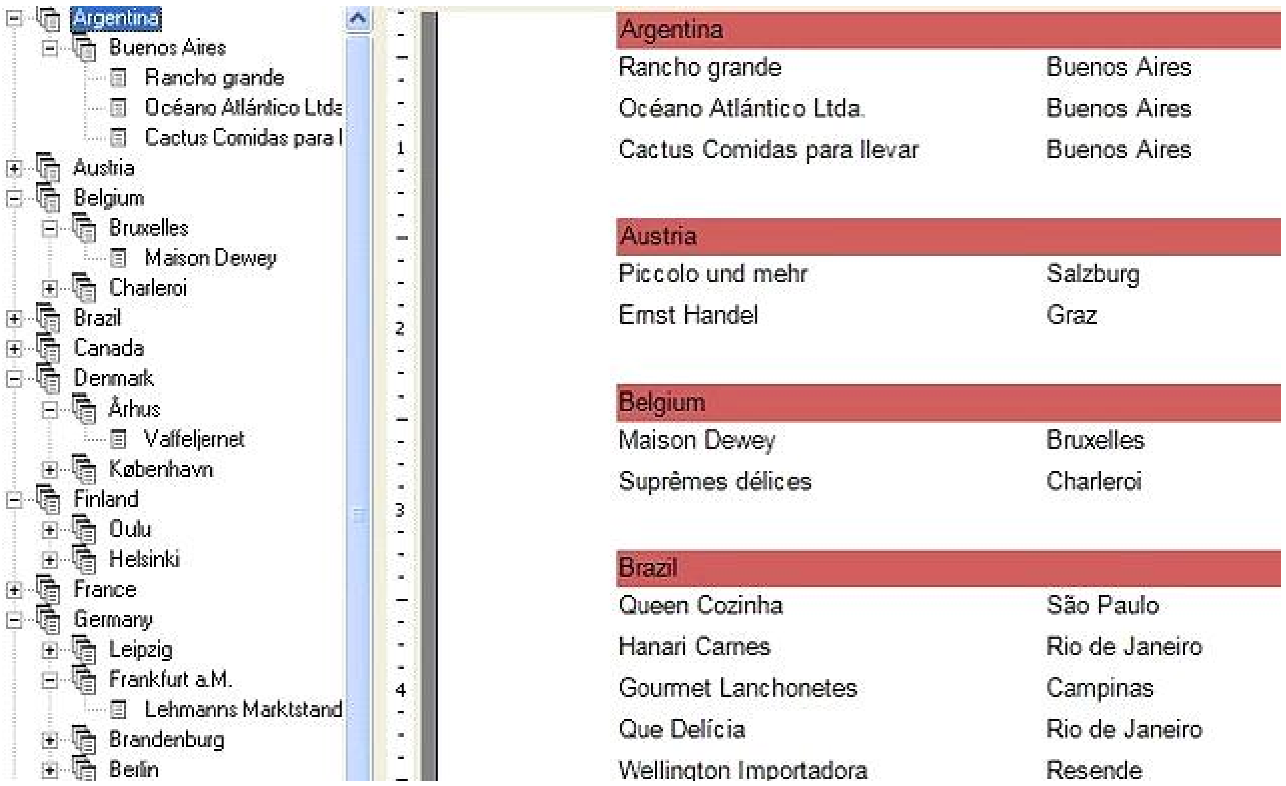
This walkthrough illustrates how to set up and use Bookmarks with grouping in a report.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Detail_Format event

- Adding code to the group header Format event
- Viewing the Bookmarks collection with the report

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb). When you have completed this walkthrough, you will have a report that looks similar to the following.



```
Argentina
  Buenos Aires
    Rancho grande
    Océano Atlántico Ltda
    Cactus Comidas para l
Austria
Belgium
  Bruxelles
    Maison Dewey
  Charleroi
Brazil
Canada
Denmark
  Århus
    Vaffeljernet
  København
Finland
  Oulu
  Helsinki
France
Germany
  Leipzig
  Frankfurt a.M.
    Lehmanns Marktstand
  Brandenburg
  Berlin
```

Argentina	
Rancho grande	Buenos Aires
Océano Atlántico Ltda.	Buenos Aires
Cactus Comidas para llevar	Buenos Aires
Austria	
Piccolo und mehr	Salzburg
Ernst Handel	Graz
Belgium	
Maison Dewey	Bruxelles
Suprêmes délices	Charleroi
Brazil	
Queen Cozinha	São Paulo
Hanari Carnes	Rio de Janeiro
Gourmet Lanchonetes	Campinas
Que Delícia	Rio de Janeiro
Wellington Importadora	Resende

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on Project > Add New Item.
3. Select **ActiveReports file** and rename the file rptGroupBM.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**.
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.

6. In the Query field, type "Select * from customers order by country".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptGroupBM
2. Make the following changes to the group header:
   - Change the name to ghCustomers
   - Change the DataField property to Country
3. Add the following control to the GroupHeader section of rptGroupBM:

Control	DataField	Name	Text/Caption	Location
TextBox	Country	**txtCountry**	Country	0, 0

4. Add the following controls to the Detail section of rptGroupBM:

Control	DataField	Name	Text/Caption	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	0, 0
TextBox	City	**txtCity**	City	2.375, 0

Adding code to the Detail_Format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptGroupBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptGroupBM, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptGroupBM's Detail_Format event.

To write the code in C#

- Click in the Detail section of rptGroupBM to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptGroupBM's Detail_Format event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e _
         As System.EventArgs) Handles Detail.Format
         Me.Detail.AddBookmark(txtCountry.Text + "\" + txtCity.Text +
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         Detail.AddBookmark(txtCountry.Text + "\\" + txtCity.Text +
}
```

Adding code to the ghCustomers_Format event

To write the code in Visual Basic

- Right-click in the GroupHeader section of the design window of rptGroupBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptGroupBM, click the drop-down arrow and select *ghCustomers*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptGroupBM's ghCustomers_Format event.

To write the code in C#

- Click in the GroupHeader section of rptGroupBM to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptGroupBM's ghCustomers_Format event.

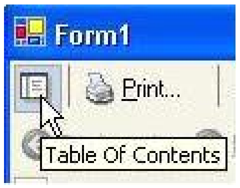
The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub ghCustomers_Format(ByVal sender As Object, ByVal e As  _
         System.EventArgs) Handles ghCustomers.Format
         Me.ghCustomers.AddBookmark(txtCountry.Text)
End Sub

[C#]

private void ghCustomers_Format(object sender, System.EventArgs eArgs)
{
         this.ghCustomers.AddBookmark(txtCountry.Text);
}
```

Viewing the Bookmarks Collection

To view the Bookmarks collection

1. Press F5 to run the report.
2. Click on the "Table of Contents" icon to view the Bookmarks collection.



Bookmarks with Simple Reports

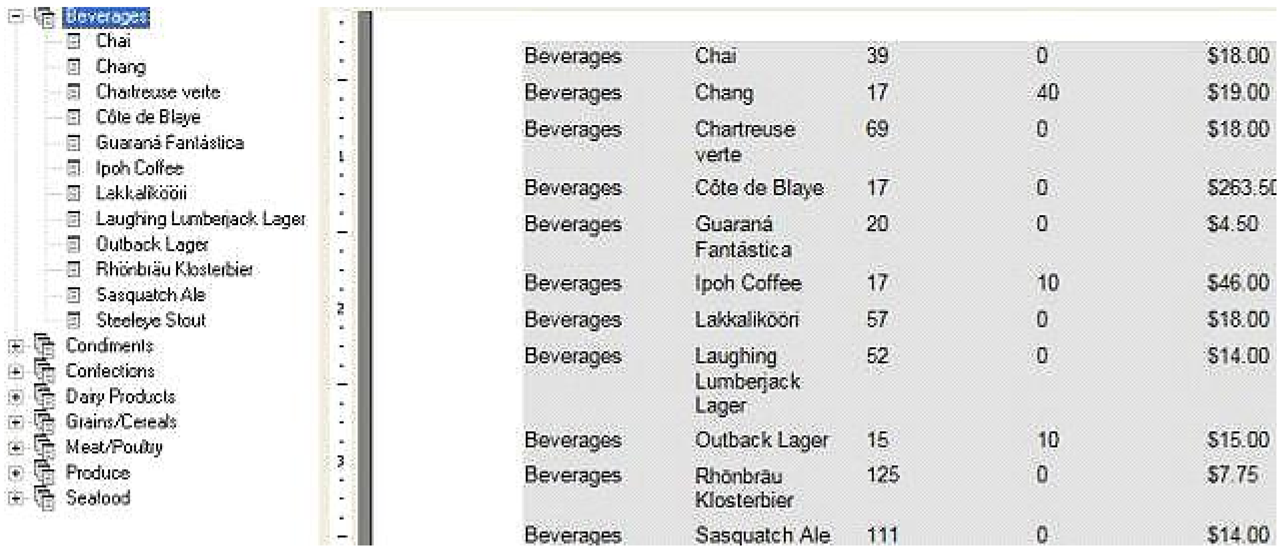
ActiveReports allows Bookmarks to be easily set up and used in simple reports by adding code to the Detail_Format event of the report.

This walkthrough illustrates how to set up and use Bookmarks in a simple report.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Detail_Format event to setup Bookmarks
- Viewing the Bookmarks collection with the report

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb). When you have completed this walkthrough, you will have a report that looks similar to the following.



Beverages
- Chai
- Chang
- Chartreuse verte
- Côte de Blaye
- Guaraná Fantástica
- Ipoh Coffee
- Lakkalikööri
- Laughing Lumberjack Lager
- Outback Lager
- Rhönbräu Klosterbier
- Sasquatch Ale
- Steeleye Stout

Condiments
Confections
Dairy Products
Grains/Cereals
Meat/Poultry
Produce
Seafood

Beverages	Chai	39	0	$18.00
Beverages	Chang	17	40	$19.00
Beverages	Chartreuse verte	69	0	$18.00
Beverages	Côte de Blaye	17	0	$263.50
Beverages	Guaraná Fantástica	20	0	$4.50
Beverages	Ipoh Coffee	17	10	$46.00
Beverages	Lakkalikööri	57	0	$18.00
Beverages	Laughing Lumberjack Lager	52	0	$14.00
Beverages	Outback Lager	15	10	$15.00
Beverages	Rhönbräu Klosterbier	125	0	$7.75
Beverages	Sasquatch Ale	111	0	$14.00

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptBmarks.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from categories inner join products on categories.categoryid = products.categoryid order by categoryname, productname".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

- Add the following controls to the Detail section of rptBmarks:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	Product Name	1.125, 0
TextBox	CategoryName	**txtCategoryName**	Category Name	0, 0
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	2.25, 0
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	3.375, 0
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4.5, 0

Adding code to the Detail_Format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptBmarks, and click on **View Code** to display the code view for the report. At the top left of the code view for rptBmarks, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptBmarks' Detail_Format event. Add code to the handler to:
    - Set up Bookmarks

To write the code in C#

- Click in the Detail section of rptBmarks to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptBmarks' Detail_Format event. Add code to the handler to:
    - Set up Bookmarks

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
         Me.Detail.AddBookmark(txtCategoryName.text)
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         Detail.AddBookmark(txtCategoryName.Text);
}
```

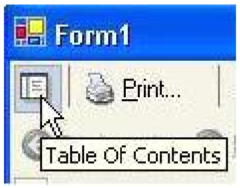
The following example shows what the code for the method looks like to set up leveled Bookmarks:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
         Me.Detail.AddBookmark(txtCategoryName.Text + "\" +
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         Detail.AddBookmark(txtCategoryName.Text + "\\" +
}
```

Viewing the Bookmarks Collection

To view the Bookmarks collection

1. Press F5 to run the report.
2. Click on the "Table of Contents" icon to view the Bookmarks collection.



Bookmarks with Subreports

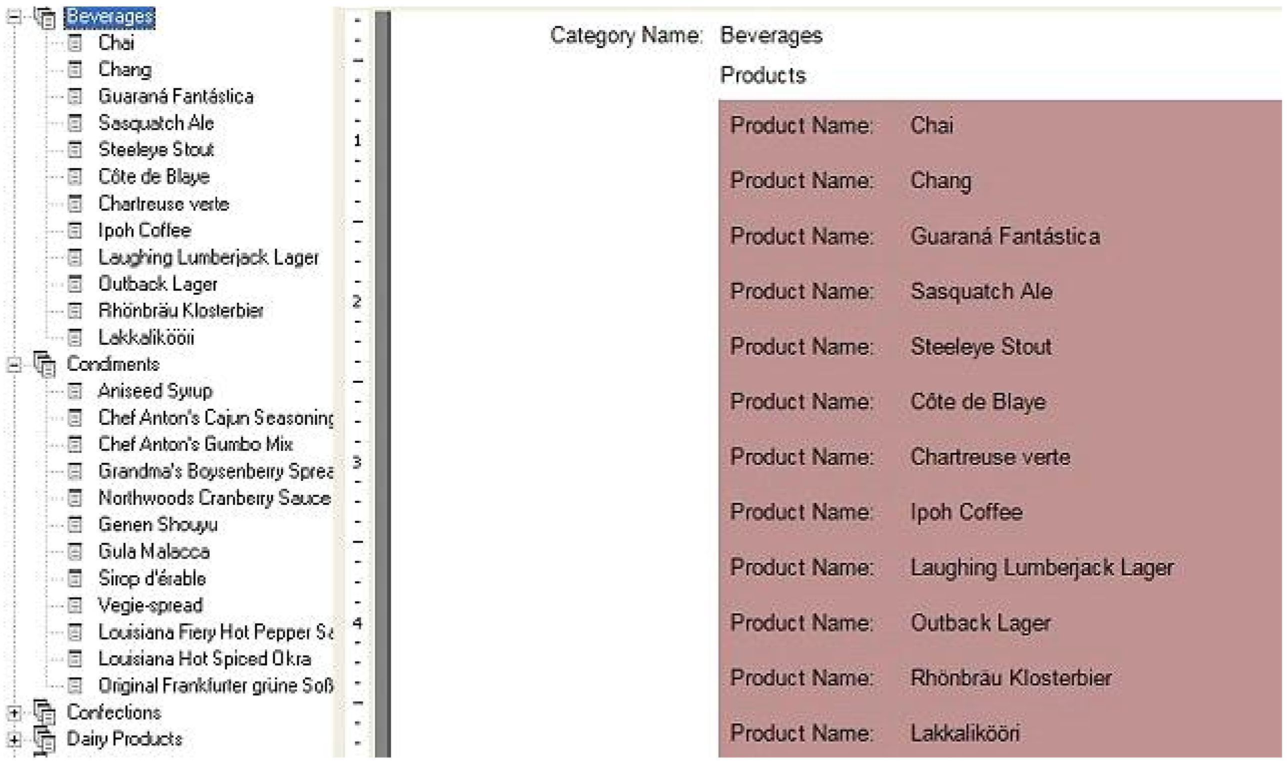
ActiveReports allows Bookmarks to be easily set up and used in subreports by adding code to the Detail_Format event of the parent and child reports.

This walkthrough illustrates how to set up and use Bookmarks in a subreport.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the parent report to a data source
- Adding controls to the report to contain data
- Adding the code needed to save the current record's CategoryID to use in the subreport's SQL query
- Adding the code to create a new data source, setting its connection string, setting its SQL query and setting the new data source equal to the subreport's data source.
- Adding code to the Detail_Format event for both reports to setup Bookmarks
- Viewing the Bookmarks collection with the report

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb). When you have completed this walkthrough, you will have a report that looks similar to the following.



```
Beverages
    Chai
    Chang
    Guaraná Fantástica
    Sasquatch Ale
    Steeleye Stout
    Côte de Blaye
    Chartreuse verte
    Ipoh Coffee
    Laughing Lumberjack Lager
    Outback Lager
    Rhönbräu Klosterbier
    Lakkalikööri
Condiments
    Aniseed Syrup
    Chef Anton's Cajun Seasonin
    Chef Anton's Gumbo Mix
    Grandma's Boysenberry Sprea
    Northwoods Cranberry Sauce
    Genen Shouyu
    Gula Malacca
    Sirop d'érable
    Vegie-spread
    Louisiana Fiery Hot Pepper S
    Louisiana Hot Spiced Okra
    Original Frankfurter grüne Soß
Confections
Dairy Products
```

```
Category Name:  Beverages
                Products
                Product Name:   Chai
                Product Name:   Chang
                Product Name:   Guaraná Fantástica
                Product Name:   Sasquatch Ale
                Product Name:   Steeleye Stout
                Product Name:   Côte de Blaye
                Product Name:   Chartreuse verte
                Product Name:   Ipoh Coffee
                Product Name:   Laughing Lumberjack Lager
                Product Name:   Outback Lager
                Product Name:   Rhönbräu Klosterbier
                Product Name:   Lakkalikööri
```

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMainBM.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptSubBM.
7. Click **Open**.

Connecting the parent report to a data source

To connect the parent report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from categories".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the reports

Add the following controls to the Detail section of rptMainBM:

Control	DataField	Name	Text/Caption	Location
Label	(Empty string)	**lblProducts**	Products	1.0625, 0.25
Label	(Empty string)	**lblCategoryName**	Category Name:	0, 0
TextBox	CategoryName	**txtCategoryName**	CategoryName	1.06, 0
Subreport	(Empty string)	**ctlSubreport**	(Empty string)	1.0625, 0.5

Add the following controls to the Detail section of rptSubBM:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	ProductName	1.187, 0.06
Label	(Empty string)	**lblProductName**	Product Name:	0.06, 0.06

Adding the code needed to save the current record's categoryID

To write the code in Visual Basic

- Right-click in any section of the design window of rptMainBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMainBM, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptMainBM's FetchData event. Add code to the handler to:
    - Save the current record's categoryID to use in the subreport's SQL query

To write the code in C#

- Click in the gray area below rptMainBM to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptMainBM's FetchData event. Add code to the handler to:
    - Save the current record's categoryID to use in the subreport's SQL query

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_categoryID As String
Private Sub rptMainBM_FetchData(ByVal sender As Object, _
         ByVal eArgs As DataDynamics.ActiveReports.ActiveReport. _
         FetchEventArgs) Handles MyBase.FetchData
         m_categoryID = Me.Fields("CategoryID").Value
End Sub

[C#]

string m_categoryID;
private void rptMainBM_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
         m_categoryID = Fields["CategoryID"].Value.ToString();
}
```

Adding the code to create a new data source

To write the code in Visual Basic

- Right-click in any section of the design surface of rptMainBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMainBM, click the drop-down arrow and select *rptMainBM*. At the top right of the code window, click the drop-down arrow and select *Detail_Format*. This creates an event-handling method for the report's Detail_Format event. Add code to the handler to:

  - Create a new DataDynamics OleDBDataSource
  - Set the new data source's connection string
  - Set the new data source's SQL query
  - Set the subreport's data source equal to the new data source

To write the code in C#

- Click in the Detail section of rptMainBM to select the section. Click on the events icon in the **Properties** window to display available events for the Detail section. Double-click *Format*. This creates an event-handling method for rptMainBM's Detail_Format event. Add code to the handler to:

  - Create a new DataDynamics OleDBDataSource
  - Set the new data source's connection string
  - Set the new data source's SQL query
  - Set the subreport's data source equal to the new data source

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
Dim rpt As New rptSubBM()
Dim subDS As New DataDynamics.ActiveReports. _
         DataSources.OleDBDataSource()
         subDS.ConnectionString = Me.ds.ConnectionString
         subDS.SQL = "Select * from products where categoryID _
                  = " + m_categoryID
         rpt.DataSource = subDS
         Me.ctlSubreport.Report = rpt
 End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         rptSub rpt = new rptSubBM();
         DataDynamics.ActiveReports.DataSources.OleDBDataSource
                  subDS = new DataDynamics.ActiveReports.
                  DataSources.OleDBDataSource();
         subDS.ConnectionString = this.ds.ConnectionString;
         subDS.SQL = "Select * from products where
                  categoryID = " + m_categoryID;
         rpt.DataSource = subDS;
         ctlSubReport.Report = rpt;
}
```

Adding code to the Detail_Format event for both reports

To write the code in Visual Basic

- Right-click in any section of the design window of rptMainBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMainBM, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptMainBM's Detail_Format event.

- Right-click in any section of the design window of rptSubBM, and click on **View Code** to display the code view for the report. At the top left of the code view for rptSubBM, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptSubBM's Detail_Format event.

To write the code in C#

- Click in the Detail section of rptMainBM to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptMainBM's Detail_Format event.

- Click in the Detail section of rptSubBM to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptSubBM's Detail_Format event.

The following example shows what the code for the method looks like for rptMainBM:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
         Me.Detail.AddBookmark(txtCategoryName.text)
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         Detail.AddBookmark(txtCategoryName.Text);
}
```

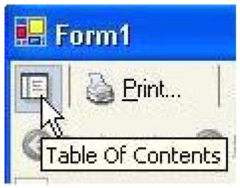
The following example shows what the code for the method looks like for rptSubBM:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
         Me.Detail.AddBookmark(CType(Me.ParentReport.Sections("Detail")._
         Controls("txtCategoryName"),  _
                  TextBox).Text + "\" + Me.txtProductName.Text)
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         this.Detail.AddBookmark(((TextBox)(this.ParentReport.Sections
                  ["Detail"].Controls["txtCategoryName"])).
                  Text + "\\" + this.txtProductName.Text);
}
```

Viewing the Bookmarks Collection

To view the Bookmarks collection

1. Press F5 to run the report.
2. Click on the "Table of Contents" icon to view the Bookmarks collection.



Calculated Fields

ActiveReports allows you to use a textbox's DataField property to perform calculations based on the value of specific data fields.

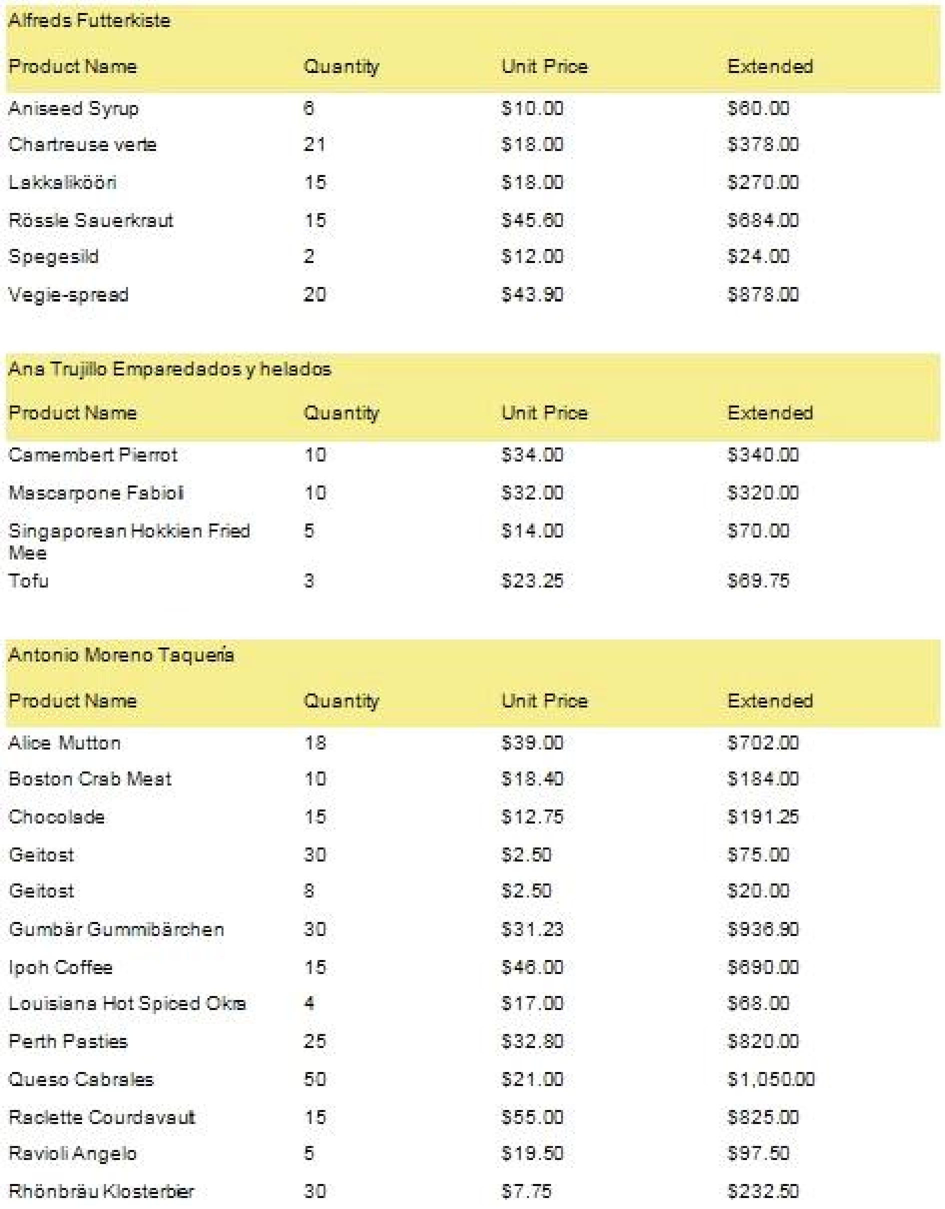
This walkthrough illustrates how to create a simple report using calculated fields.

The walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Alfreds Futterkiste			
Product Name	Quantity	Unit Price	Extended
Aniseed Syrup	6	$10.00	$60.00
Chartreuse verte	21	$18.00	$378.00
Lakkalikööri	15	$18.00	$270.00
Rössle Sauerkraut	15	$45.60	$684.00
Spegesild	2	$12.00	$24.00
Vegie-spread	20	$43.90	$878.00

Ana Trujillo Emparedados y helados			
Product Name	Quantity	Unit Price	Extended
Camembert Pierrot	10	$34.00	$340.00
Mascarpone Fabioli	10	$32.00	$320.00
Singaporean Hokkien Fried Mee	5	$14.00	$70.00
Tofu	3	$23.25	$69.75

Antonio Moreno Taquería			
Product Name	Quantity	Unit Price	Extended
Alice Mutton	18	$39.00	$702.00
Boston Crab Meat	10	$18.40	$184.00
Chocolade	15	$12.75	$191.25
Geitost	30	$2.50	$75.00
Geitost	8	$2.50	$20.00
Gumbär Gummibärchen	30	$31.23	$936.90
Ipoh Coffee	15	$46.00	$690.00
Louisiana Hot Spiced Okra	4	$17.00	$68.00
Perth Pasties	25	$32.80	$820.00
Queso Cabrales	50	$21.00	$1,050.00
Raclette Courdavault	15	$55.00	$825.00
Ravioli Angelo	5	$19.50	$97.50
Rhönbräu Klosterbier	30	$7.75	$232.50

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptCalFields.
4. Click **Open**.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select Customers.CompanyName, Products.ProductName, [Order Details].UnitPrice, [Order Details].Quantity FROM Products INNER JOIN ((Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID) INNER JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID) ON Products.ProductID = [Order Details].ProductID WHERE (((DatePart("yyyy", [OrderDate])) = 1995)) ORDER BY Customers.CompanyName, Products.ProductName".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptCalFields.
2. Make the following changes to the group header:
   - o Change the name to ghProducts
   - o Change the DataField to CompanyName
3. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	0, 0
Label	(Empty string)	**lblProductName**	Product Name	0, 0.3125
Label	(Empty string)	**lblUnitPrice**	Unit Price	3.437, 0.3125
Label	(Empty string)	**lblQuantity**	Quantity	2.0625, 0.3125
Label	(Empty string)	**lblExtended**	Extended	5, 0.3125

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	Quantity	**txtQuantity**	Quantity	2.0625, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	3.437, 0	Currency
TextBox	= Quantity * UnitPrice	**txtExtended**	Extended	5, 0	Currency

Conditional Formatting

ActiveReports allows you to modify or suppress the appearance of any control at run time based on conditions in your data. This can be achieved by setting properties of the control in the section's format event based on certain conditions.

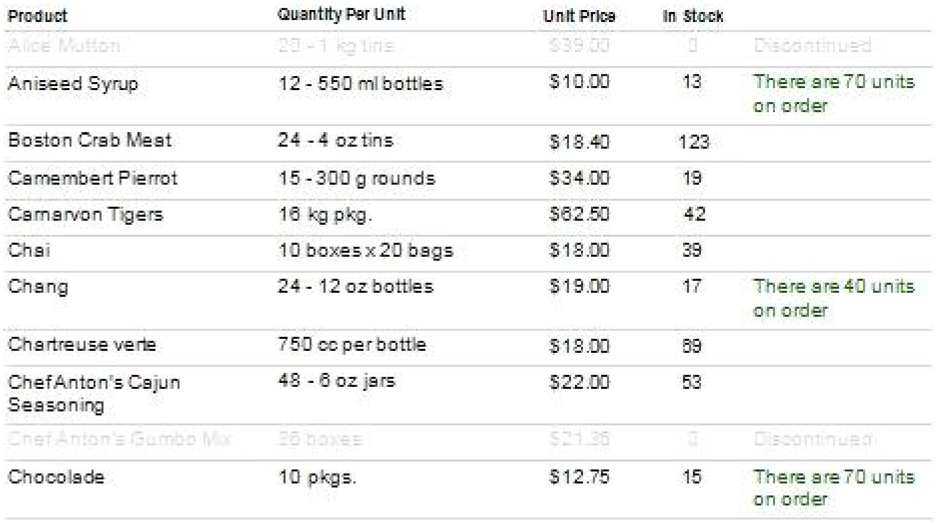
This walkthrough illustrates how to create a report based on specific conditions that modify the appearance of the report at run time.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding conditions in code to the format event
- Adding code to retrieve data from the data source

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product	Quantity Per Unit	Unit Price	In Stock	
Alice Mutton	20 - 1 kg tins	$39.00	0	Discontinued
Aniseed Syrup	12 - 550 ml bottles	$10.00	13	There are 70 units on order
Boston Crab Meat	24 - 4 oz tins	$18.40	123	
Camembert Pierrot	15 - 300 g rounds	$34.00	19	
Carnarvon Tigers	16 kg pkg.	$62.50	42	
Chai	10 boxes x 20 bags	$18.00	39	
Chang	24 - 12 oz bottles	$19.00	17	There are 40 units on order
Chartreuse verte	750 cc per bottle	$18.00	69	
Chef Anton's Cajun Seasoning	48 - 6 oz jars	$22.00	53	
Chef Anton's Gumbo Mix	36 boxes	$21.35	0	Discontinued
Chocolade	10 pkgs.	$12.75	15	There are 70 units on order

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptConFormat.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products ORDER BY productname".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add the following controls to the PageHeader section:

Control	Name	Text/Caption	Location
Label	**lblProduct**	Product	0, 0
Label	**lblQuantityPerUnit**	Quantity Per Unit	1.375, 0
Label	**lblUnitPrice**	Unit Price	2.6875, 0
Label	**lblInStock**	In Stock	4.0625, 0

2. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	QuantityPerUnit	**txtQuantityPerUnit**	Quantity Per Unit	1.375, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	2.687, 0	Currency
TextBox	UnitsInStock	**txtInStock**	Units In Stock	4.062, 0	(Empty string)

Adding conditions in code to the format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptConFormat, and click on **View Code** to display the code view for the report. At the top left of the code view for rptConFormat, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptConFormat's Detail_Format event. Add code to the handler to:
  - Set conditions in the format event

To write the code in C#

- Click in the Detail section of rptConFormat to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptConFormat's Detail_Format event. Add code to the handler to:
  - Set conditions in the format event

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_uis As Integer
Dim m_rl As Integer
Dim m_uoo As Integer
Dim m_dis As Boolean
```

```
Private Sub Detail_Format(ByVal sender As Object, ByVal e As _     System.EventArgs)
Handles Detail.Format
         If m_uis < m_rl Then
                  If m_uoo = 0 Then
                           Me.txtWarning.Text = "Time to Reorder"
                           txtWarning.ForeColor = System.Drawing.Color.Red
                  ElseIf m_uoo > 0 Then
                           txtWarning.Text = "There are " + _
                                    m_uoo.ToString() + "units _
                                    on order"
                           txtWarning.ForeColor = System.Drawing. _
                                    Color.DarkGreen
                  End If
         Else
                  txtWarning.Text = ""
                  txtWarning.ForeColor = System.Drawing.Color.Black
         End If
         If (m_dis) Then
                  Me.txtInStock.ForeColor = System.Drawing.Color.LightGray
                  Me.txtProductName.ForeColor = System.Drawing. _
                           Color.LightGray
                  Me.txtQuantityPerUnit.ForeColor = System.Drawing. _
                           Color.LightGray
                  Me.txtUnitPrice.ForeColor = System.Drawing. _
                           Color.LightGray
                  Me.txtWarning.ForeColor = System.Drawing.Color.LightGray
                  Me.txtWarning.Text = "Discontinued"
         Else
                  Me.txtInStock.ForeColor = System.Drawing.Color.Black
                  Me.txtProductName.ForeColor = System.Drawing.Color.Black
                  Me.txtQuantityPerUnit.ForeColor = System.Drawing. _
                           Color.Black
                  Me.txtUnitPrice.ForeColor = System.Drawing.Color.Black
         End If
End Sub

[C#]

int m_uis;
int m_rl;
int m_uoo;
bool m_dis;
private void Detail_Format(object sender, System.EventArgs eArgs)
{
         if(m_uis < m_rl)
         {
                  if(m_uoo == 0)
                  {
                           txtWarning.Text= "Time To Reorder";
                           txtWarning.ForeColor = System.Drawing.Color.Red;
                  }
                  else
                  {
                           txtWarning.Text = "There are " + m_uoo.ToString()
                           txtWarning.ForeColor = System.Drawing.
                                    Color.DarkGreen;
                  }
         }
         else
         {
                  txtWarning.Text ="";
                  txtWarning.ForeColor = System.Drawing.Color.Black;
         }
         if(m_dis)
         {
                  this.txtInStock.ForeColor = System.Drawing.
                           Color.LightGray;
                  this.txtProduct.ForeColor = System.Drawing.
                           Color.LightGray;
                  this.txtQuantityPerUnit.ForeColor = System.Drawing.
                           Color.LightGray;
```

```
                    this.txtUnitPrice.ForeColor = System.Drawing.
                              Color.LightGray;
                    this.txtWarning.ForeColor = System.Drawing.
                              Color.LightGray;
                    this.txtWarning.Text = "Discontinued";
          }
          else
          {
                    this.txtInStock.ForeColor = System.Drawing.Color.Black;
                    this.txtProduct.ForeColor = System.Drawing.Color.Black;
                    this.txtQuantityPerUnit.ForeColor = System.Drawing.
                              Color.Black;
                    this.txtUnitPrice.ForeColor = System.Drawing.
                              Color.Black;
          }
}
```

Adding code to retrieve data from the data source

To write the code in Visual Basic

- Right-click in any section of the design window of rptConFormat, and click on **View Code** to display the code view for the report. At the top left of the code view for rptConFormat, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptConFormat's FetchData event. Add code to the handler to:

    - Retrieve information from the data source

To write the code in C#

- Click in the gray area below rptConFormat to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptConFormat's FetchData event. Add code to the handler to:

    - Retrieve information from the data source

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptConFormat_FetchData(ByVal sender As Object, _
          ByVal eArgs As DataDynamics.ActiveReports. _
          ActiveReport.FetchEventArgs) Handles MyBase.FetchData
          m_uis = Fields("UnitsInStock").Value
          m_rl = Fields("ReorderLevel").Value
          m_uoo = Fields("UnitsOnOrder").Value
          m_dis = Fields("Discontinued").Value
End Sub

[C#]

private void rptConFormat_FetchData(object sender,
          DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
          m_uis = (System.Int16)Fields["UnitsInStock"].Value;
          m_rl = (System.Int16)Fields["ReorderLevel"].Value;
          m_uoo = (System.Int16)Fields["UnitsOnOrder"].Value;
          m_dis = (System.Boolean)Fields["Discontinued"].Value;
}
```

Custom Exporting Walkthroughs

ActiveReports provides custom components for several formats, including PDF, HTML, RTF, Excel and plain text. Ultimate customizability is available by using any ASP.NET language. The following walkthroughs demonstrate how to set up report custom exporting to HTML and PDF.

- Custom Exporting with HTML
- Custom Exporting with PDF

HTML

ActiveReports provides custom components for several formats, including PDF, HTML, RTF, Excel and plain text. Ultimate customizability is available by using any ASP.NET language.

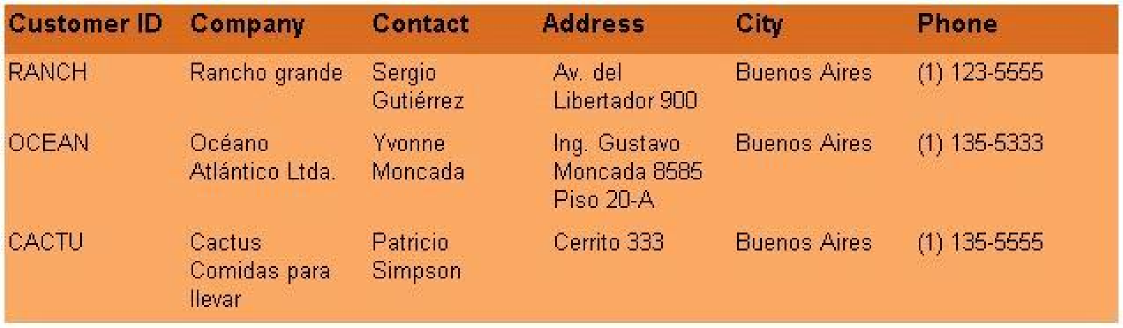
This walkthrough illustrates how to create a simple Web application and set up custom exporting in HTML format.

This walkthrough is split up into the following activities:

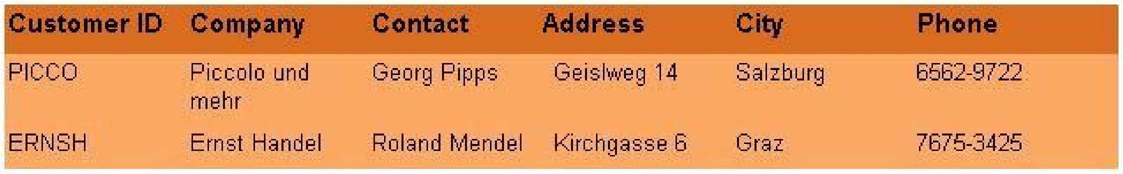
- Adding an ActiveReport to an ASP.NET Web application
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Web Form to export to HTML
- Creating a public class for the HTML outputter
- Adding a folder to the project for report output

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb). You must also have access to Internet Information Services either from your computer or from the server. You must also run the "Configure Web Sample" option from the Data Dynamics ActiveReports for .NET program menu from your Windows Start button.

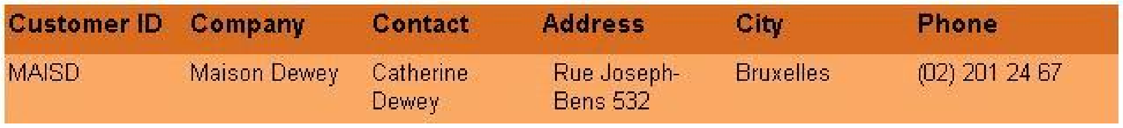
When you have completed this walkthrough, you will have a report that looks similar to the following.



Customer ID	Company	Contact	Address	City	Phone
RANCH	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	(1) 123-5555
OCEAN	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	(1) 135-5333
CACTU	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	(1) 135-5555



Customer ID	Company	Contact	Address	City	Phone
PICCO	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	6562-9722
ERNSH	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	7675-3425



Customer ID	Company	Contact	Address	City	Phone
MAISD	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	(02) 201 24 67

Adding an ActiveReport to an ASP.NET Web application

To add an ActiveReport to your project

1. Open a new ASP.NET Web application in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptCustExHTML.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers order by country".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptCustExHTML.
2. Make the following changes to the group header:

- Change the name to ghCountries
- Change the DataField property to Country

3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblCustomerID**	Customer ID	0, 0
Label	**lblCompanyName**	Company Name	1.0625, 0
Label	**lblContactName**	Contact Name	2.125, 0
Label	**lblAddress**	Address	3.125, 0
Label	**lblCity**	City	4.25, 0
Label	**lblPhone**	Phone	5.3125, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	CustomerID	**txtCustomerID**	Customer ID	0, 0
TextBox	CompanyName	**txtCompanyName**	Company Name	1.0625, 0
TextBox	ContactName	**txtContactName**	Contact Name	2.125, 0
TextBox	Address	**txtAddress**	Address	3.125, 0
TextBox	City	**txtCity**	City	4.25, 0
TextBox	Phone	**txtPhone**	Phone	5.3125, 0

Adding code to the Web Form to export to HTML

To write the code in Visual Basic

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

To write the code in C#

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Private Sub Page_Load(ByVal sender As System.Object, ByVal _
         e As System.EventArgs)Handles MyBase.Load
        Dim rpt As New ActiveReport1()
        Try
            rpt.Run(False)
        Catch eRunReport As Exception
            ' Failure running report, just report the
               'error to the user:
            Response.Clear()
            Response.Write("<h1>Error running report:</h1>")
            Response.Write(eRunReport.ToString())
            Return
        End Try
        ' Buffer this page's output until the report output is ready.
        Response.Buffer = True
           ' Clear any part of this page that might have already been
           '  buffered for output.
        Response.ClearContent()
           ' Clear any headers that might have already been buffered ' '
           ' (such as the content type for an HTML page)
        Response.ClearHeaders()
        ' Tell the browser and the "network" that this resulting data
           ' of this page should be be cached since
        ' this could be a dynamic report that changes on each request.
        Response.Cache.SetCacheability(HttpCacheability.NoCache)
        ' Tell the browser this is a PDF document so it will use an
          ' appropriate viewer.
        Response.ContentType = "text/HTML"
        ' Create the HTML export object
        Dim html As New DataDynamics.ActiveReports.Export.HtmlExport()
        ' Export the report to HTML in this session's webcache:
        'html.Export(rpt.Document, DataDynamics.ActiveReports.Web
        Dim outputter As New MyCustomHtmlOutputter(Me.Context)
        html.Export(rpt.Document, outputter, "")
        Response.Redirect("ReportOutput" + "/" + _
                   System.IO.Path.GetFileName(outputter.mainPage))
End Sub

[C#]

private void Page_Load(object sender, System.EventArgs e)
{
         rptCustExHTML rpt = new rptCustExHTML();
         try
         {
                  rpt.Run(false);
         }
         catch (Exception eRunReport)
         {
                  // Failure running report, just report the error to the
                  // user:
                  Response.Clear();
                  Response.Write("<h1>Error running report:</h1>");
                  Response.Write(eRunReport.ToString());
                  return;
         }
         // Buffer this page's output until the report output is ready.
         Response.Buffer = true;
         // Clear any part of this page that might have already been
         // buffered for output.
         Response.ClearContent();
         // Clear any headers that might have already been buffered (such
         //as the content type for an HTML page)
         Response.ClearHeaders();
         // Tell the browser and the "network" that this resulting data
         // of this page should be be cached since this
         // could be a dynamic report that changes upon each request.
         Response.Cache.SetCacheability(HttpCacheability.NoCache);
         // Tell the browser this is a PDF document so it will use an
         // appropriate viewer.
         Response.ContentType = "application/pdf";
         // Create the HTML export object
         DataDynamics.ActiveReports.Export.HtmlExport html = new
                  DataDynamics.ActiveReports.Export.HtmlExport();
         // Export the report to HTML in this session's webcache:
         // html.Export(rpt.Document, DataDynamics.ActiveReports.Web
         // MyCustomHtmlOutputter outputter = new
         // MyCustomHtmlOutputter(this.Context);
         html.Export(rpt.Document, outputter, "");
         Response.Redirect("ReportOutput" + "/" + _
                  System.IO.Path.GetFileName(outputter.mainPage));
}
```

Creating a public class for the HTML outputter

To create a public class

1. In the Solution Explorer window, right-click on your project name.
2. Click **Add**, **Add New Item**.
3. In the **Templates** window of the **Add New Item** dialog, click "Class".
4. Change the name of the file to "MyCustomHtmlOutputter" and click **Open**.
5. This will open the code view of the class file where you can add the code needed to create the public class.

To write the code in Visual Basic

- Add the following code between "Public Class MyCustomHtmlOutputter" and "End Class".

To write the code in C#

- Add the following code under "public class MyCustomHtmlOutputter".

The following example shows what the complete code for the method looks like:

```
[Visual Basic]

 Public Class MyCustomHtmlOutputter
    Implements DataDynamics.ActiveReports.Export.Html.IOutputHtml
    ' The http context of the request.
    Dim context
    ' The directory the filename will be saved in--this will be used to
    ' ensure the filename is unique.
    Dim dirToSave
    Public mainPage As String = ""
    Public Sub New(ByVal context As System.Web.HttpContext)
        MyBase.New()
        If context Is Nothing Then
            Throw New ArgumentNullException("context")
        End If
        Me.context = context
        Dim dirName As String = context.Server.MapPath("ReportOutput")
        Me.dirToSave = New DirectoryInfo(dirName)
End Sub

#Region "Implementation of IOutputHtml"
Public Function OutputHtmlData(ByVal info As _
         DataDynamics.ActiveReports.Export.Html.HtmlOutputInfoArgs) As  _
         String Implements IOutputHtml.OutputHtmlData
        Dim temp As String = ""
        Select Case info.OutputKind
            Case HtmlOutputKind.BookmarksHtml
            Case HtmlOutputKind.FramesetHtml
                temp = Me.GenUniqueFileNameWithExtension(".html")
                Dim fs As New FileStream(temp, FileMode.CreateNew)
                Me.WriteStreamToStream(info.OutputStream, fs)
                fs.Close()
                Return temp
            Case HtmlOutputKind.HtmlPage
                ' We want to hold on to the name of the main page so we                '
can redirect the browser to it
                Me.mainPage = Me.GenUniqueFileNameWithExtension _
                            (".html")
                Dim fs As New FileStream(Me.mainPage, _
                            FileMode.CreateNew)
                Me.WriteStreamToStream(info.OutputStream, fs)
                fs.Close()
                Return Me.mainPage
            Case HtmlOutputKind.ImageJpg
```

```
                ' should be a file with a .jpg extension:
                temp = Me.GenUniqueFileNameWithExtension(".jpg")
                Dim fs As New FileStream(temp, FileMode.CreateNew)
                fs = File.Create(temp)
                Me.WriteStreamToStream(info.OutputStream, fs)
                fs.Close()
                Return temp
            Case HtmlOutputKind.ImagePng
                ' should be a file with a .png extension:
                temp = Me.GenUniqueFileNameWithExtension(".png")
                Dim fs As New FileStream(temp, FileMode.CreateNew)
                Me.WriteStreamToStream(info.OutputStream, fs)
                fs.Close()
                Return temp
            Case Else
                ' This case shouldn't really happen, but we'll default
                    ' to html.
                temp = Me.GenUniqueFileNameWithExtension(".html")
                Dim fs As New FileStream(temp, FileMode.CreateNew)
                Me.WriteStreamToStream(info.OutputStream, fs)
                fs.Close()
                Return temp
        End Select
End Function
Public Sub Finish() Implements IOutputHtml.Finish
End Sub
#End Region

Private Sub WriteStreamToStream(ByVal sourceStream As Stream, _
         ByVal targetStream As Stream)
        ' What's the size of the source stream:
        Dim size As Integer = CType(sourceStream.Length, Integer)
        ' Create a buffer that same size:
        Dim buffer(size) As Byte
        ' Move the source stream to the beginning:
        sourceStream.Seek(0, SeekOrigin.Begin)
        'copy the whole sourceStream into our buffer:
        sourceStream.Read(buffer, 0, size)
        'Write out buffer to the target stream:
        targetStream.Write(buffer, 0, size)
End Sub
Private Function GenUniqueFileNameWithExtension(ByVal  _
         extensionWithDot As String) As String
        Dim r As New System.Random()
        Dim unique As Boolean = False
        Dim filePath As String = ""
        Dim iRandom As Integer = 0
        ' Generate a random name until it's unique:
        While Not unique
            iRandom = r.Next()
            ' Buld the full filename
            Dim sb = New StringBuilder()
            sb.Append(Me.dirToSave.FullName)
            sb.Append(Path.DirectorySeparatorChar)
            sb.Append(iRandom.ToString())
            sb.Append(extensionWithDot)
            filePath = sb.ToString()
            If File.Exists(filePath) = False Then
                unique = True
            Else
                unique = False
            End If
        End While
        Return filePath
End Function
End Class

[C#]

public class MyCustomHtmlOutputter:DataDynamics.ActiveReports.Export.Html.
         IOutputHtml
```

```
{
          // The http context of the request.
          private System.Web.HttpContext context = null;
          // The directory the filename will be saved in--this will be
          // used to ensure the filename is unique.
          private System.IO.DirectoryInfo dirToSave = null;
          public string mainPage = "";
          public MyCustomHtmlOutputter(System.Web.HttpContext context)
{
                    if (context == null)
                              throw new ArgumentNullException("context");
                    this.context = context;
                    string dirName = context.Server.MapPath("ReportOutput");
                    this.dirToSave = new DirectoryInfo(dirName);
}
#region Implementation of IOutputHtml

public string OutputHtmlData(DataDynamics.ActiveReports.Export.Html.
          HtmlOutputInfoArgs info)
{
          string temp = "";
          switch (info.OutputKind)
          {
                    case HtmlOutputKind.BookmarksHtml:
                    case HtmlOutputKind.FramesetHtml:
                    {
                              temp = this.GenUniqueFileNameWithExtension
                                        (".html");
                              FileStream fs = File.Create(temp);
                              this.WriteStreamToStream(info.OutputStream, fs);
                              fs.Close();
                              return temp;
                    }
                    case HtmlOutputKind.HtmlPage:
                    {
                              // We want to hold on to the name of the main
                              // page so we can redirect the browser to it:
                              this.mainPage =
                                        this.GenUniqueFileNameWithExtension
                                        (".html");
                              FileStream fs = File.Create(this.mainPage);
                              this.WriteStreamToStream(info.OutputStream, fs);
                              fs.Close();
                              return this.mainPage;
                    }
                    case HtmlOutputKind.ImageJpg:
                    {
                              // should be a file with a .jpg extension:
                              temp = this.GenUniqueFileNameWithExtension
                                        (".jpg");
                              FileStream fs = File.Create(temp);
                              this.WriteStreamToStream(info.OutputStream, fs);
                              fs.Close();
                              return temp;
                    }
                    case HtmlOutputKind.ImagePng:
                    {
                              // should be a file witha .png extension:
                              temp = this.GenUniqueFileNameWithExtension
                                        (".png");
                              FileStream fs = File.Create(temp);
                              this.WriteStreamToStream(info.OutputStream, fs);
                              fs.Close();
                              return temp;
                    }
                    default:
                    {
                              // This case shouldn't really happen, but we'll
                              // default to html.
                              temp = this.GenUniqueFileNameWithExtension
                                        (".html");
```

```
                                FileStream fs = File.Create(temp);
                                this.WriteStreamToStream(info.OutputStream, fs);
                                fs.Close();
                                return temp;
                     }
           }
}

public void Finish()
{
}
```

Adding a folder to the project for report output

To add a folder to the project

1. Open Windows Explorer and browse to the folder in which your project is contained.
2. On the **File** menu, click **New**, **Folder**.
3. Name the folder "ReportOutput".
4. Make sure that you have write permissions for this folder.

PDF

ActiveReports provides custom components for several formats, including PDF, HTML, RTF, Excel and plain text. Ultimate customizability is available by using any ASP.NET language.

This walkthrough illustrates how to create a simple Web application and set up custom exporting in PDF format.

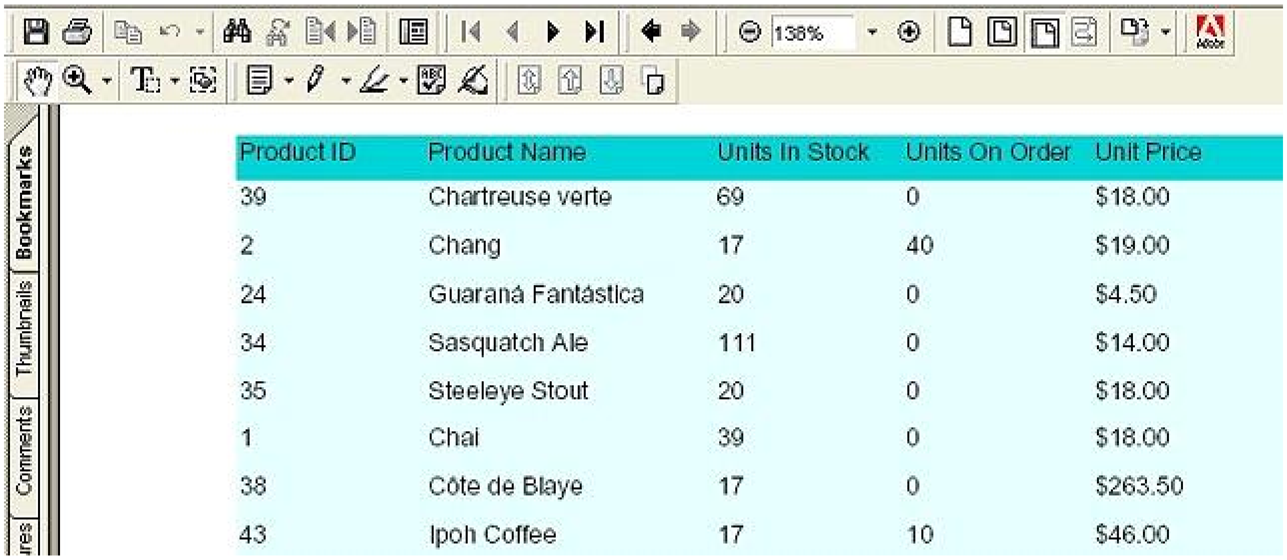
This walkthrough is split up into the following activities:

- Adding an ActiveReport to an ASP.NET Web application
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Web Form to export to PDF

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

You must also have access to Internet Information Services either from your computer or from the server. You must also run the "Configure Web Sample" option from the Data Dynamics ActiveReports for .NET program menu from your Windows Start button.

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product ID	Product Name	Units In Stock	Units On Order	Unit Price
39	Chartreuse verte	69	0	$18.00
2	Chang	17	40	$19.00
24	Guaraná Fantástica	20	0	$4.50
34	Sasquatch Ale	111	0	$14.00
35	Steeleye Stout	20	0	$18.00
1	Chai	39	0	$18.00
38	Côte de Blaye	17	0	$263.50
43	Ipoh Coffee	17	10	$46.00

Adding an ActiveReport to an ASP.NET Web application

To add an ActiveReport to your project

1. Open a new ASP.NET Web application in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptCustEx.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products order by categoryID".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptCustEx.
2. Make the following changes to the group header:
   - o Change the name to ghProducts
   - o Change the DataField property to CategoryID
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location

Label	**lblProductID**	Product ID	0, 0
Label	**lblProductName**	Product Name	1.0625, 0
Label	**lblUnitsInStock**	Units In Stock	2.6875, 0
Label	**lblUnitsOnOrder**	Units On Order	3.75, 0
Label	**lblUnitPrice**	Unit Price	4.8125, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductID	**txtProductID**	Product ID	0, 0	(Empty string)
TextBox	ProductName	**txtProductName**	Product Name	1.0625, 0	(Empty string)
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	2.6875, 0	(Empty string)
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	3.75, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4.8125, 0	Currency

Adding code to the Web Form to export to PDF

To write the code in Visual Basic

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

To write the code in C#

- Double-click on WebForm1. This creates an event-handling method for WebForm1's Page_Load event. Add the following code to the Page_Load event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Page_Load(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles MyBase.Load
        Dim p As New DataDynamics.ActiveReports.Export.Pdf.PdfExport()
        Dim m_stream As New System.IO.MemoryStream()
        Dim rpt As New rptCustEx()

        rpt.Run()
        p.Export(rpt.Document, m_stream)
        m_stream.Position = 0
        Response.ContentType = "application/pdf"
        Response.BinaryWrite(m_stream.ToArray())
        Response.End()
End Sub

[C#]

private void Page_Load(object sender, System.EventArgs e)
{
         DataDynamics.ActiveReports.Export.PdfExport p = new
                  DataDynamics.ActiveReports.Export.Pdf.PdfExport();
         System.IO.MemoryStream m_stream = new System.IO.MemoryStream();
         rptCustEx rpt = new rptCustEx();
```

```
          rpt.Run();
          p.Export(rpt.Document, m_stream);
          m_stream.Position = 0;
          Response.ContentType = "application/pdf";
          Response.BinaryWrite(m_stream.ToArray());
          Response.End();
}
```

Customizing the Viewer Control

ActiveReports includes a control to view report output in custom preview forms or in Microsoft Internet Explorer. The viewer allows developers to modify the toolbars or add custom menu commands to the preview form.

This walkthrough illustrates how to add and customize the ActiveReports viewer control to your report.

This walkthrough is split up into the following activities:

- Creating a custom preview screen
- Using split windows on the viewer control
- Adding a button to the viewer control

To complete the walkthrough, you must have the ActiveReports controls added to your Visual Studio toolbox. For more information, see "Adding ActiveReports Controls to the Visual Studio Toolbox."

Creating a custom preview screen in a report

To create a custom preview screen in a report

1. Open an ActiveReport in Visual Studio.
2. Add a Form to your project..
3. Click on the Viewer icon on the ActiveReports toolbox.

4. Place the control on your form and size it according to your needs.
5. Set the Dock property to Fill.
6. Add the following code to the Form1_Load event:

```
[Visual Basic]

Dim rpt as new ActiveReport1
Viewer1.Document = rpt.Document
rpt.Run()

[C#]

    ActiveReport1 rpt = new ActiveReport1();
    viewer1.Document = rpt.Document;
    rpt.Run();
```

7. Press F5 to run the report.

Using split windows on the viewer control

To use split windows on the viewer control

1. Run your report with the viewer added by pressing F5.
2. Drag the splitter control down.

3. When the viewer is split into two sections, report layouts can be examined and report pages can be compared easily.

Adding a button to the viewer control

To add a button to the viewer control

1. Open an ActiveReport in Visual Studio.
2. Add a form to your project.
3. Set up the ActiveReports viewer on Form1 following the steps outlined above.
4. Add a second form to your project and rename it frmPrintDlg.
5. Add a label to frmPrintDlg and change the Text property to "This is the custom print dialog."
6. Add a button to frmPrintDlg and change the Text property to "OK". (This button is for appearance only in this walkthrough.)
7. Add the following code to the Form1_Load event:

```
[Visual Basic]

' Remove the default print button
Me.Viewer1.Toolbar.Tools.RemoveAt(2)

' Create and add the custom button
Dim btn As New DataDynamics.ActiveReports.Toolbar.Button()
btn.Caption = "MyPrint"
btn.ToolTip = "Custom Print Button"
btn.ImageIndex = 1
btn.ButtonStyle = DataDynamics.ActiveReports.Toolbar.ButtonStyle.TextAndIcon
btn.Id = 333
Me.Viewer1.Toolbar.Tools.Insert(2, btn)

[C#]

// Remove the default printer button
this.viewer1.Toolbar.Tools.RemoveAt(2);

// Create and add the custom button
DataDynamics.ActiveReports.Toolbar.Button btn = new
         DataDynamics.ActiveReports.Toolbar.Button();
btn.Caption = "MyPrint";
btn.ToolTip = "Custom Print Button";
btn.ImageIndex = 1;
btn.ButtonStyle =  DataDynamics.ActiveReports.Toolbar.ButtonStyle.TextAndIcon;
btn.Id = 333;
this.viewer1.Toolbar.Tools.Insert(2,btn);
```

8. Add the following code to the Viewer1_ToolClick event:

```
[Visual Basic]

' Capture the new tool's click to show the dialog
If e.Tool.Id = 333 Then
         Dim dlg As New frmPrintDlg()
         dlg.ShowDialog(Me)
End If

[C#]

// Capture the new tool's click to show the dialog
if(e.Tool.Id == 333)
{
         frmPrintDlg dlg = new frmPrintDlg();
         dlg.ShowDialog(this);
```

Data Bound Reports

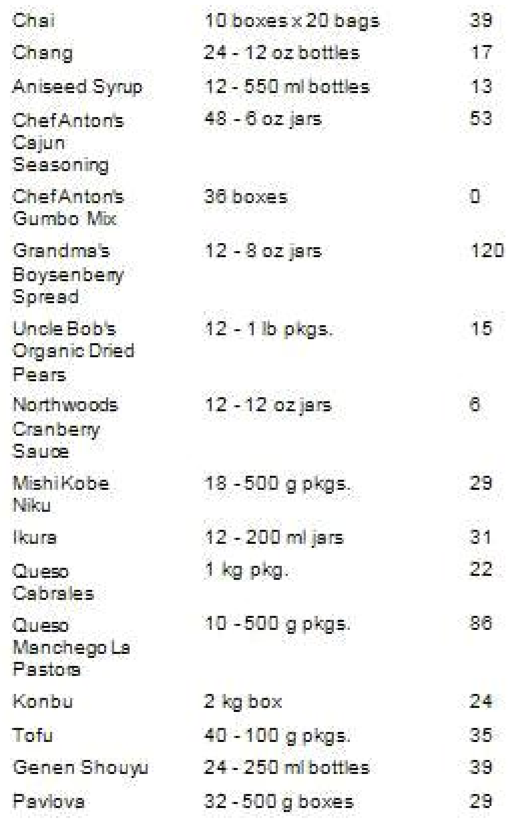
In ActiveReports, the simplest reporting style is a tabular listing of fields from a record source. This walkthrough illustrates the basics of setting up bound reports by introducing the ideas of using the DataSource icon and connecting textbox controls to the data source through the DataField property.

The walkthrough is split up into the following activities:

- Creating a new Visual Studio project
- Adding an ActiveReport to the Visual Studio project
- Connecting the data source to a database
- Adding controls to contain the data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Chai	10 boxes x 20 bags	39
Chang	24 - 12 oz bottles	17
Aniseed Syrup	12 - 550 ml bottles	13
Chef Anton's Cajun Seasoning	48 - 6 oz jars	53
Chef Anton's Gumbo Mix	36 boxes	0
Grandma's Boysenberry Spread	12 - 8 oz jars	120
Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	15
Northwoods Cranberry Sauce	12 - 12 oz jars	6
Mishi Kobe Niku	18 - 500 g pkgs.	29
Ikura	12 - 200 ml jars	31
Queso Cabrales	1 kg pkg.	22
Queso Manchego La Pastora	10 - 500 g pkgs.	86
Konbu	2 kg box	24
Tofu	40 - 100 g pkgs.	35
Genen Shouyu	24 - 250 ml bottles	39
Pavlova	32 - 500 g boxes	29

Creating a new Visual Studio project

To create a new Visual Studio project

1. Open Visual Studio.
2. Click on **Open New Project** or click on **File > New > Project**.
3. Select the project type and click on **Windows Application**.
4. Change the name of your project and click **OK**.

Adding an ActiveReport to the Visual Studio project

To add an ActiveReport to your project

1. Click on **Project > Add New Item**.
2. Select **ActiveReports file** and rename the file rptBound.
3. Click **Open**.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to contain data

1. Drag the following fields from the Report Explorer window: ProductName, QuantityPerUnit and UnitsInStock.
2. Drop these textboxes into the Detail section and arrange them horizontally in the above order.
3. Resize the Detail section to remove extra white space.

Deploying Compiled Reports

With ActiveReports, compiled reports can be set up for deployment by including the ActiveReports deployment .msm file in your Visual Studio deployment project.

This walkthrough illustrates how to create a deployment project in ActiveReports for a compiled report.

This walkthrough is split up into the following activities:

- Adding an installer project to an existing ActiveReports project
- Adding the ActiveReports .msm file
- Adding the ActiveReports application to the installer
- Deploying the installer application

Adding an installer project to an existing ActiveReports project

To add the installer project

1. Open an existing ActiveReports project or create a new report.
2. On the **Build** menu, click "Build [your ActiveReports project name]" to build your report project.
3. On the **File** menu, select **Add Project** and click on **New Project...**
4. Under Project Types in the Add New Project dialog, select Setup and Deployment Projects.
5. In the **Templates** window, select **Setup Project,** rename the file and click **OK**.
6. Select the Installer project in Solution Explorer. In the Properties window, select the **ProductName** property and type in the name of your file.

   Note The **ProductName** property determines the name that will be displayed for the application in folder names and in the **Add/Remove Programs** dialog box.

Adding the ActiveReports .msm file

To add the ActiveReports .msm file

1. Right-click on the Installer project in Solution Explorer.
2. Click on **Add** and then click **Merge Module...**
3. Open the Deployment folder where ActiveReports is installed (e.g. c:\\program files\Data Dynamics\ActiveReports.NET\Deployment).
4. Click on "ActiveReportsDistrib.msm" to select it and click **Open**.
5. This adds all of the ActiveReports distributed assemblies to your project.

   Note Since the Setup and Deployment project will automatically detect and add any existing assembly dependencies to your project and the .msm file adds all ActiveReports assemblies, you will need to exclude any duplicate ActiveReports DLLs from the "Detected Dependencies" folder in the Solution Explorer window.

Adding the ActiveReports application to the installer

To add the ActiveReports application

1. Select the Installer project in Solution Explorer.
2. In the File System Editor, choose the Application folder.
3. On the Action menu, select **Add**, **Project Output...**
4. In the **Add Project Output Group** dialog, choose your ActiveReports project name from the drop-down list.
5. Select "Primary Output" from the list and click **OK**.
6. On the **Build** menu, click "Build [your Installer project name]" to build your Installer project.

Deploying the installer application

To deploy the installer application

1. Select the Installer project in Solution Explorer.
2. On the **Project** menu, click **Install**.
3. The Installer application will run and install the project on your computer.

Exporting Output

Included with ActiveReports are several specialized export filters (HTML, PDF, RTF, Excel, TIFF and Text). With these export filters, reports easily can be made available to others in various formats.

This walkthrough illustrates how to export a report to PDF.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Referencing the appropriate export filter in your project
- Connecting the report to a data source
- Adding controls to display the data
- Using the export method to export data to PDF

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have created a PDF file which can be found in the Bin subfolder of your project's folder.

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptExports.
4. Click **Open**.

Referencing the export filter in your project

To reference the export filter in your project

1. Click anywhere on the report to select it.
2. Click on **Project > Add Reference...**
3. Under the .NET tab, click on the appropriate reference. For this walkthrough, choose "Data Dynamics ActiveReports PDF Export Filter." Other available references include:
   - ARExportExcel
   - ARExportTiff
   - Data Dynamics ActiveReports HTML Export Filter
   - Data Dynamics ActiveReports Rich Text Format Export Filter
   - Data Dynamics ActiveReports Text Export Filter
4. Click on **Select** > **OK**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from employees".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the report

- Add the following controls to the Detail section:

Control	DataField	Name	Text	Location
TextBox	LastName	**txtLastName**	Last Name	0.0625, 0.0625
TextBox	FirstName	**txtFirstName**	First Name	1.1875, 0.0625

Using the export method to export files to PDF

To write the code in Visual Basic

- Right-click on Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
    - Export the report to a PDF file

To write the code in C#

- Click on the blue section at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for Form1. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
    - Export the report to a PDF file

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Viewer1_Load(ByVal sender As System.Object, _
         ByVal e As System.EventArgs)Handles Viewer1.Load
Dim pExp As New DataDynamics.ActiveReports.Export.Pdf.PdfExport()
         pExp.Export(Viewer1.Document, Application.StartupPath _
                   + "\\PDFExpt.PDF")
End Sub

[C#]
```

```
private void Form1_Load(object sender, System.EventArgs e)
{
         DataDynamics.ActiveReports.Export.PDF.PdfExport pExp = new
                  DataDynamics.ActiveReports.Export.Pdf.PdfExport();
         pExp.Export(viewer1.Document, Application.StartupPath +
                  "\\PDFExpt.PDF");
}
```

Grouping Data Walkthroughs

With ActiveReports, grouping can be easily added to your reports. The following walkthroughs describe exactly how to include various grouping options in your report.

- Conditional Show/Hide Detail
- Group on Simple Fields
- Group on Unbound Fields
- Keeptogether Options

Conditional Show/Hide Detail

ActiveReports allows you to hide or show information from the data source in the Detail section of your report based on conditions in your data. This can be achieved by setting properties of the section in the Format event based on certain conditions.

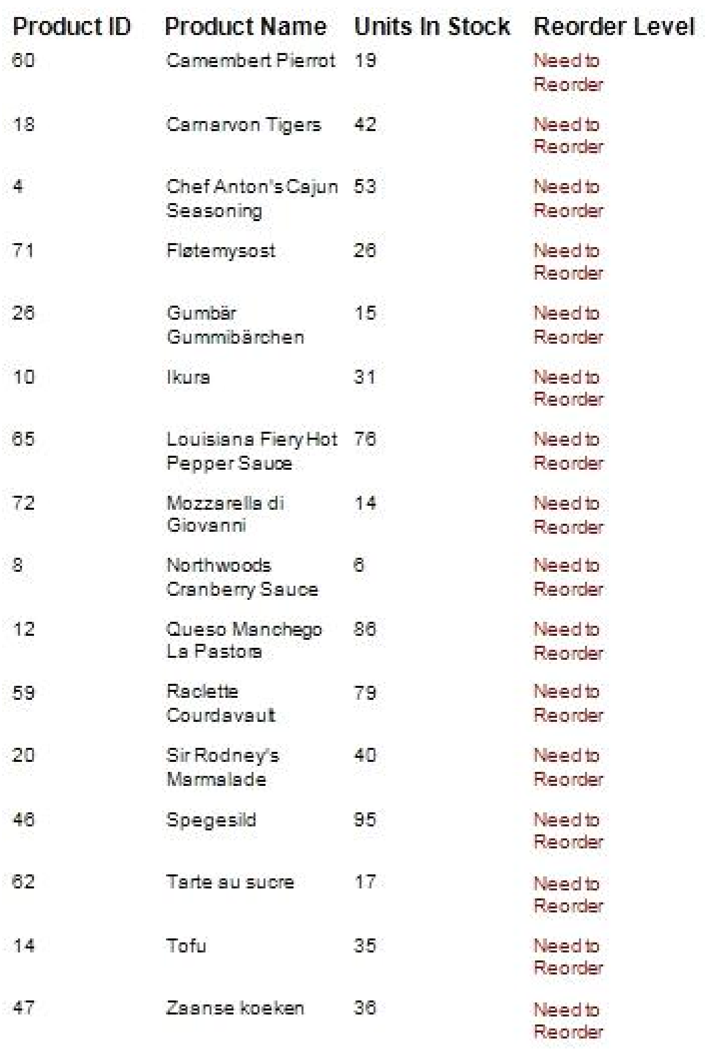
This walkthrough illustrates how to create a report based on conditions that will show specific data from your data source at run time.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding conditions in code to the Format event
- Adding code to retrieve data from the data source

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product ID	Product Name	Units In Stock	Reorder Level
60	Camembert Pierrot	19	Need to Reorder
18	Carnarvon Tigers	42	Need to Reorder
4	Chef Anton's Cajun Seasoning	53	Need to Reorder
71	Fløtemysost	26	Need to Reorder
26	Gumbär Gummibärchen	15	Need to Reorder
10	Ikura	31	Need to Reorder
65	Louisiana Fiery Hot Pepper Sauce	76	Need to Reorder
72	Mozzarella di Giovanni	14	Need to Reorder
8	Northwoods Cranberry Sauce	6	Need to Reorder
12	Queso Manchego La Pastora	86	Need to Reorder
59	Raclette Courdavault	79	Need to Reorder
20	Sir Rodney's Marmalade	40	Need to Reorder
46	Spegesild	95	Need to Reorder
62	Tarte au sucre	17	Need to Reorder
14	Tofu	35	Need to Reorder
47	Zaanse koeken	36	Need to Reorder

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptCondSH.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.

5. Click **OK** to continue.
6. In the Query field, type "Select * from products ORDER BY productname".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a PageHeader/Footer section to the report.
2. Add the following controls to the PageHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductID**	Product ID	0, 0
Label	**lblProductName**	Product Name	1.0625, 0
Label	**lblReorderLevel**	Reorder Level	3.8125, 0
Label	**lblInStock**	Units In Stock	2.5, 0

3. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductID	**txtProductID**	Product ID	0, 0
TextBox	ProductName	**txtProductName**	Product Name	1.0625, 0
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	2.510, 0
TextBox	ReorderLevel	**txtReorderLevel**	Reorder Level	3.833, 0
TextBox	Discontinued	**txtDiscontinued**	Discontinued	5.291, 0

Adding conditions in code to the format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptCondSH, and click on **View Code** to display the code view for the report. At the top left of the code view for rptCondSH, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptCondSH's Detail_Format event. Add code to the handler to:

  - Set conditions in the Format event

To write the code in C#

- Click in the Detail section of rptCondSH to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptCondSH's Detail_Format event. Add code to the handler to:

  - Set conditions in the Format event

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_rl As Integer
Dim m_dis As Boolean
Private Sub Detail_Format(ByVal sender As Object, ByVal e As _
          System.EventArgs) Handles Detail.Format
          If m_rl = 0 And m_dis = False Then
                    Me.Detail.Visible = True
                    Me.txtDiscontinued.Text = ""
                    Me.txtReorderLevel.Text = "Need to Reorder"
                    Me.txtReorderLevel.ForeColor = System.Drawing. _
                              Color.DarkRed
          Else
                    Me.Detail.Visible = False
          End If
End Sub

[C#]

int m_rl;
bool m_dis;
private void Detail_Format(object sender, System.EventArgs eArgs)
{
          if(m_rl == 0 && m_dis == false)
          {
                    this.Detail.Visible = true;
                    this.txtDiscontinued.Text = "";
                    this.txtReorderLevel.Text = "Need to Reorder";
                    this.txtReorderLevel.ForeColor = System.Drawing. _
                              Color.DarkRed;
          }
          else
          {
                    this.Detail.Visible = false;
          }
}
```

Adding code to retrieve data from the data source

To write the code in Visual Basic

- Right-click in any section of the design window of rptCondSH, and click on **View Code** to display the code view for the report. At the top left of the code view for rptCondSH, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptCondSH's FetchData event. Add code to the handler to:
  - Retrieve information from the data source

To write the code in C#

- Click in the gray area below rptCondSH to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptCondSH's FetchData event. Add code to the handler to:
  - Retrieve information from the data source

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptCondSH_FetchData(ByVal sender As Object,  _
          ByVal eArgs As DataDynamics.ActiveReports.ActiveReport. _
          FetchEventArgs) Handles MyBase.FetchData
```

```
  m_rl = Fields("ReorderLevel").Value
  m_dis = Fields("Discontinued").Value
End Sub

[C#]

private void rptCondSH_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
         m_rl = (System.Int16)Fields["ReorderLevel"].Value;
         m_dis = (System.Boolean)Fields["Discontinued"].Value;
}
```

Group on Simple Fields

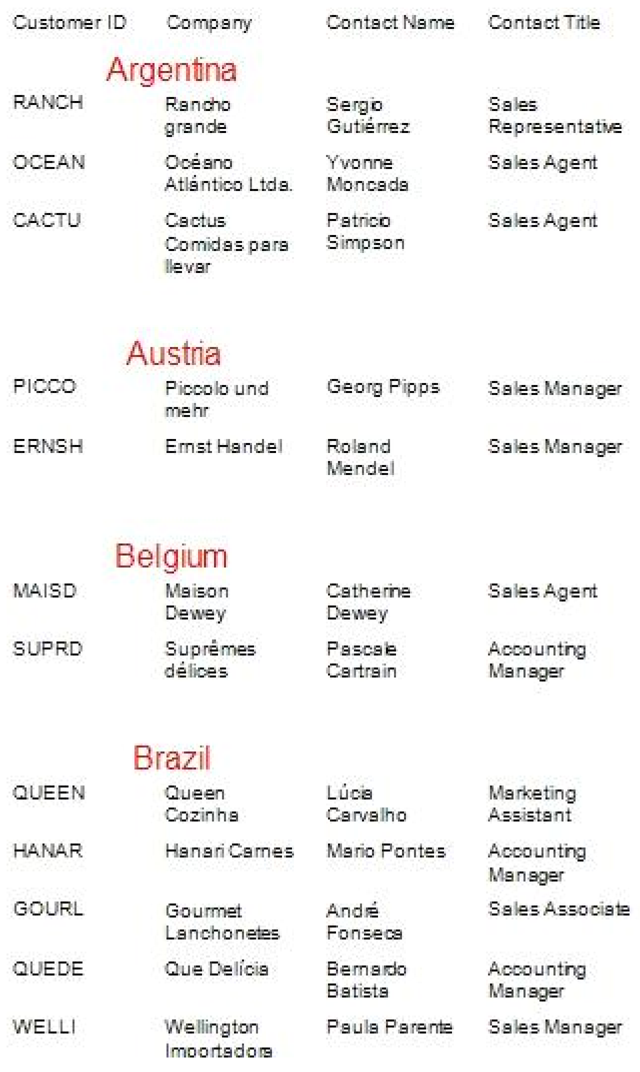
In ActiveReports, reports can be grouped by using a group header with the DataField property set to the database field on which it is being grouped. ActiveReports allows up to 32 nested groups in a single report. When using grouping, make sure the returned data set is ordered by the fields on which it is being grouped.

This walkthrough illustrates the basics of setting up grouping on simple fields in a report.

- The walkthrough is split up into the following activities:
- Adding an ActiveReport to your project
- Connecting the data source to a database
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Customer ID	Company	Contact Name	Contact Title
Argentina			
RANCH	Rancho grande	Sergio Gutiérrez	Sales Representative
OCEAN	Océano Atlántico Ltda.	Yvonne Moncada	Sales Agent
CACTU	Cactus Comidas para llevar	Patricio Simpson	Sales Agent
Austria			
PICCO	Piccolo und mehr	Georg Pipps	Sales Manager
ERNSH	Ernst Handel	Roland Mendel	Sales Manager
Belgium			
MAISD	Maison Dewey	Catherine Dewey	Sales Agent
SUPRD	Suprêmes délices	Pascale Cartrain	Accounting Manager
Brazil			
QUEEN	Queen Cozinha	Lúcia Carvalho	Marketing Assistant
HANAR	Hanari Carnes	Mario Pontes	Accounting Manager
GOURL	Gourmet Lanchonetes	André Fonseca	Sales Associate
QUEDE	Que Delícia	Bernardo Batista	Accounting Manager
WELLI	Wellington Imoortadora	Paula Parente	Sales Manager

Adding an ActiveReport to your project

To add an ActiveReport to your project

1. Click on **Project > Add New Item....**
2. Click on **ActiveReports File** to select it.
3. Change the name of the report to rptSimpleGroup and click **Open**.
4. The ActiveReports design surface is displayed.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail field. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.

5. Click **OK** to continue.
6. In the Query field, type "Select * from customers ORDER BY country".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to your report (see Grouping Data for help).
2. Make the following changes to the group header:
   - o Change the name to ghOrderGroup
   - o Change the DataField property to Country
3. Add the following controls to the PageHeader section:

Control	Name	Text/Caption	Location
Label	**lblCustomerID**	Customer ID	0, 0
Label	**lblCompanyName**	Company Name	1.0729, 0
Label	**lblContactName**	Contact Name	2.1875, 0
Label	**lblContactTitle**	Contact Title	3.3125, 0

4. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Location
Textbox	Country	**txtCountry**	Country	0.3125, 0.0625

5. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
Textbox	CustomerID	**txtCustomerID**	Customer ID	0, 0
Textbox	CompanyName	**txtCompanyName**	Company Name	1.0625, 0
Textbox	ContactName	**txtContactName**	Contact Name	2.1875, 0
Textbox	ContactTitle	**txtContactTitle**	Contact Title	3.3125, 0

Group on Unbound Fields

ActiveReports allows you to set up grouping in unbound reports. When setting up grouping, the group header's DataField property is used in the same manner as a textbox's DataField property to retrieve the grouping data from the database.

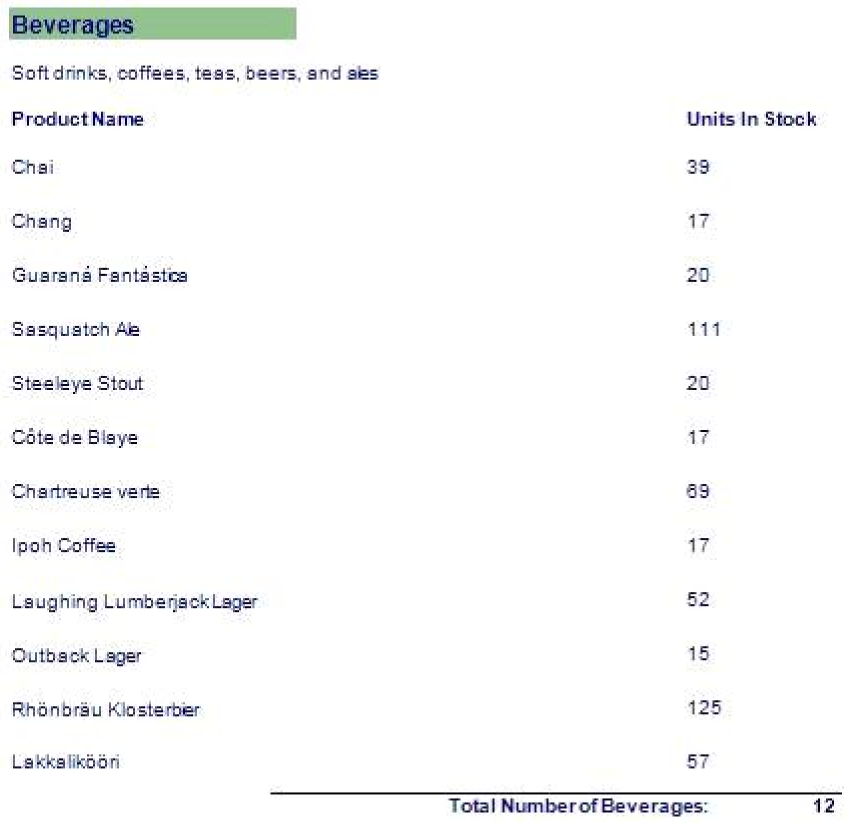
This walkthrough illustrates how to set up grouping in an unbound report.

This walkthrough is split into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Adding code to connect the report to a data source
- Adding controls to contain the data
- Using the DataInitialize event to add fields to the report's fields collection
- Using the FetchData event to populate the report fields

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Beverages

Soft drinks, coffees, teas, beers, and ales

Product Name	Units In Stock
Chai	39
Chang	17
Guaraná Fantástica	20
Sasquatch Ale	111
Steeleye Stout	20
Côte de Blaye	17
Chartreuse verte	69
Ipoh Coffee	17
Laughing Lumberjack Lager	52
Outback Lager	15
Rhönbräu Klosterbier	125
Lakkalikööri	57

Total Number of Beverages: 12

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptUnboundGrp.
4. Click **Open**.

Adding code to connect the report to a data source

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnboundGrp, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnboundGrp, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for rptUnboundGrp's ReportStart event. Add code to the handler to:

  - Set the data source connection string
  - Set the data source SQL query
  - Open the connection to create the data reader

To write the code in C#

- Click in the gray area below rptUnboundGrp to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for rptUnboundGrp's FetchData event. Add code to the handler to:

  - Set the data source connection string
  - Set the data source SQL query
  - Open the connection to create the DataReader

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_cnnString As String
Dim sqlString As String
Dim m_reader As OleDbDataReader
Dim m_cnn As OleDbConnection
Private Sub rptUnboundGrp_ReportStart(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles MyBase.ReportStart
  m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data _
         Source=C:\Program Files\Data Dynamics\ActiveReports.NET _
         \Data\NWIND.MDB;Persist Security Info=False"
  sqlString = "SELECT * FROM categories INNER JOIN products  _
         ON categories.categoryid = products.categoryid  _
         ORDER BY products.categoryid, products.productid"
  m_cnn = New OleDb.OleDbConnection(m_cnnString)
  Dim m_Cmd As New OleDb.OleDbCommand(sqlString, m_cnn)
  If m_cnn.State = ConnectionState.Closed Then
    m_cnn.Open()
  End If
  m_reader = m_Cmd.ExecuteReader()
End Sub

[C#]

private static OleDbConnection m_cnn;
private static OleDbDataReader  m_reader;

private void rptUnboundGrp_ReportStart(object sender, System.EventArgs
         eArgs)
{
         string m_dbPath = getDatabasePath();
         string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
                  Source=C:\Program Files\Data
                  Dynamics\ActiveReports.NET\Data\NWIND.MDB;Persist
                  Security Info=False";
         string sqlString = "SELECT * FROM categories INNER JOIN products            ON
categories.categoryid = products.
                  categoryid ORDER BY products.categoryid,
                  products.productid";
         m_cnn = new OleDbConnection(m_cnnString);
         OleDbCommand m_Cmd = new OleDbCommand(sqlString,m_cnn);
         if(m_cnn.State == ConnectionState.Closed)
         {
                  m_cnn.Open();
         }
         m_reader = m_Cmd.ExecuteReader();
}
```

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to your report (see Grouping Data for help).
2. Make the following changes to the group header:
   - Change the name to ghCategories
   - Change the DataField property to CategoryID
   - Set the GroupKeepTogether property to All
   - Set the KeepTogether property to True
3. Make the following change to the group footer:
   - Change the name to gfCategories
4. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Location
TextBox	CategoryName	**txtCategoryName**	Category Name	0.0625, 0.0625
TextBox	Description	**txtDescription**	Description	0.0625, 0.375
Label	(Empty string)	**lblProductName**	Product Name	0.0625, 0.6875
Label	(Empty string)	**lblUnitsInStock**	Units In Stock	4.75, 0.6875

5. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	Product Name	0.0625, 0.0625
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	4.75, 0.0625

6. Add the following controls to the GroupFooter section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	TotalLabel	**lblTotalLabel**	Total Label	(Empty string)	1.875, 0
TextBox	ProductName	**txtTotalItems**	Total Items	Summary Type = Subtotal SummaryFunc = Count SummaryRunning = Group SummaryGroup = ghCategories	5, 0
Line	(Empty string)	**Line1**	(Empty string)	LineWeight = 3	X1 = 1.875 Y1 = 0 X2 = 6.4375 Y2 = 0

Using the DataInitialize event to add fields

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnboundGrp, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnboundGrp, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *DataInitialize*. This creates an event-handling method for rptUnboundGrp's DataInitialize event. Add code to the handler to:
  - Add fields to the report's fields collection.

To write the code in C#

- Click in the gray area below rptUnboundGrp to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *DataInitialize*. This creates an event-handling method for rptUnboundGrp's DataInitialize event. Add code to the handler to:
    - Add fields to the report's fields collection.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptUnboundGrp_DataInitialize(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.DataInitialize
  Fields.Add("CategoryID")
  Fields.Add("CategoryName")
  Fields.Add("ProductName")
  Fields.Add("UnitsInStock")
  Fields.Add("Description")
  Fields.Add("TotalLabel")
End Sub

[C#]

private void UnboundGrp_DataInitialize(object sender, System.EventArgs
         eArgs)
{
         Fields.Add("CategoryID");
         Fields.Add("CategoryName");
         Fields.Add("ProductName");
         Fields.Add("UnitsInStock");
         Fields.Add("Description");
         Fields.Add("TotalLabel");
}
```

Using the FetchData event to populate the report fields

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnboundGrp, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnboundGrp, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptUnboundGrp's FetchData event. Add code to the handler to:
    - Retrieve information to populate the report fields.

To write the code in C#

- Click in the gray area below rptUnboundGrp to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptUnboundGrp's FetchData event. Add code to the handler to:
    - Retrieve information to populate the report fields.

The following example shows what the code for the method looks like:

```
[Visual Basic]
```

```
Private Sub rptUnboundGrp_FetchData(ByVal sender As Object, _
         ByVal eArgs As DataDynamics.ActiveReports.ActiveReport. _
         FetchEventArgs) Handles MyBase.FetchData
  Try
    m_reader.Read()
    Me.Fields("CategoryID").Value = m_reader("categories.CategoryID")
    Me.Fields("CategoryName").Value = m_reader("CategoryName")
    Me.Fields("ProductName").Value = m_reader("ProductName")
    Me.Fields("UnitsInStock").Value = m_reader("UnitsInStock")
    Me.Fields("Description").Value = m_reader("Description")
    Me.Fields("TotalLabel").Value = "Total Number of " + _
         m_reader("CategoryName") + ":"
    eArgs.EOF = False
  Catch ex As Exception
    System.Windows.Forms.MessageBox.Show(ex.ToString())
    eArgs.EOF = True
  End Try
End Sub

[C#]

private void UnboundGrp_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
         try
         {
                  m_reader.Read();
                  Fields["CategoryID"].Value =
                           m_reader["categories.CategoryID"].
                           ToString();
                  Fields["CategoryName"].Value =
                           m_reader["CategoryName"].ToString();
                  Fields["ProductName"].Value =
                           m_reader["ProductName"].ToString();
                  Fields["UnitsInStock"].Value =
                           m_reader["UnitsInStock"].ToString();
                  Fields["Description"].Value =
                           m_reader["Description"].ToString();
                  Fields["TotalLabel"].Value = "Total Number of " +
                           m_reader["CategoryName"].ToString() + ":";
                  eArgs.EOF = false;
         }
         catch
         {
                  eArgs.EOF = true;
         }
}
```

Keeptogether Options

ActiveReports allows you to set Keeptogether options for your reports so that group detail is kept together on one page when printed.

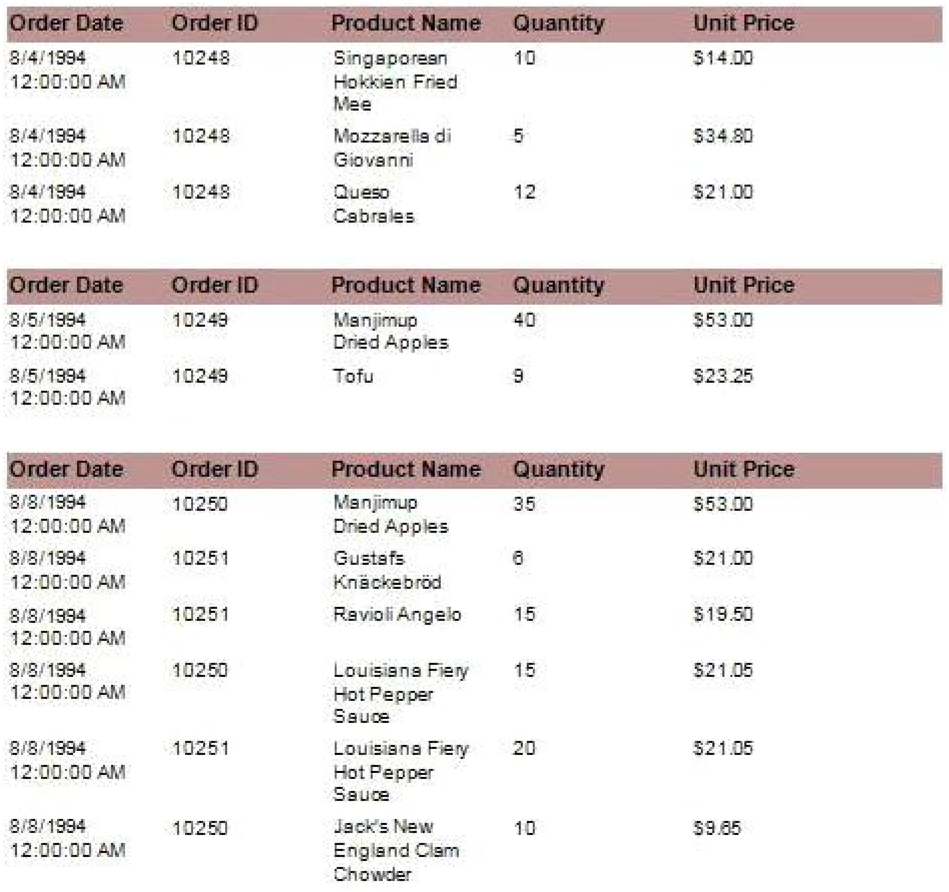
This walkthrough illustrates how to set the Keeptogether and GroupKeepTogether options to allow all group detail to print together on one page.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data.

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Order Date	Order ID	Product Name	Quantity	Unit Price
8/4/1994 12:00:00 AM	10248	Singaporean Hokkien Fried Mee	10	$14.00
8/4/1994 12:00:00 AM	10248	Mozzarella di Giovanni	5	$34.80
8/4/1994 12:00:00 AM	10248	Queso Cabrales	12	$21.00

Order Date	Order ID	Product Name	Quantity	Unit Price
8/5/1994 12:00:00 AM	10249	Manjimup Dried Apples	40	$53.00
8/5/1994 12:00:00 AM	10249	Tofu	9	$23.25

Order Date	Order ID	Product Name	Quantity	Unit Price
8/8/1994 12:00:00 AM	10250	Manjimup Dried Apples	35	$53.00
8/8/1994 12:00:00 AM	10251	Gustafs Knäckebröd	6	$21.00
8/8/1994 12:00:00 AM	10251	Ravioli Angelo	15	$19.50
8/8/1994 12:00:00 AM	10250	Louisiana Fiery Hot Pepper Sauce	15	$21.05
8/8/1994 12:00:00 AM	10251	Louisiana Fiery Hot Pepper Sauce	20	$21.05
8/8/1994 12:00:00 AM	10250	Jack's New England Clam Chowder	10	$9.65

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptKeepTG.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.

6. In the Query field, type "SELECT DISTINCTROW Orders.*, [Order Details].*, Products.* FROM Products INNER JOIN (Orders INNER JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID) ON Products.ProductID = [Order Details].ProductID order by orderdate".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptKeepTG (see Grouping Data for help).
2. Make the following changes to the group header:
   - Change the name to ghOrders
   - Change the DataField to OrderDate
   - Change the GroupKeepTogether property to All
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblOrderDate**	Order Date	0, 0
Label	**lblOrderID**	Order ID	1.125, 0
Label	**lblProductName**	Product Name	2.239, 0
Label	**lblQuantity**	Quantity	3.5, 0
Label	**lblUnitPrice**	Unit Price	4.75, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	orders.OrderID	**txtOrderID**	Order ID	1.125, 0	(Empty string)
TextBox	ProductName	**txtProductName**	Product Name	2.260, 0	(Empty string)
TextBox	products.UnitPrice	**txtUnitPrice**	Unit Price	4.75, 0	Currency
TextBox	OrderDate	**txtOrderDate**	Order Date	0, 0	(Empty string)
TextBox	Quantity	**txtQuantity**	Quantity	3.5, 0	(Empty string)

Hyperlinks Walkthroughs

ActiveReports allows hyperlinks to be added to reports. These reports can then be previewed, displayed in the viewer control or exported. The hyperlink property can be set to any HTML-style link, used to simulate drill-down reporting or set to items in the Table of Contents. By using the hyperlink property, reports can have "clickable" controls which can be used for a variety of different tasks, including running and displaying other reports. The following walkthroughs demonstrate the different ways hyperlinking can be used in reports.

- Hyperlinks
- Hyperlinks and Bookmarks
- Hyperlinks and Simulated Drill-Down Reporting

Hyperlinks

ActiveReports allows you to add hyperlinks to reports. The hyperlink property can be set to any HTML-style link such as http:// and mailto://.

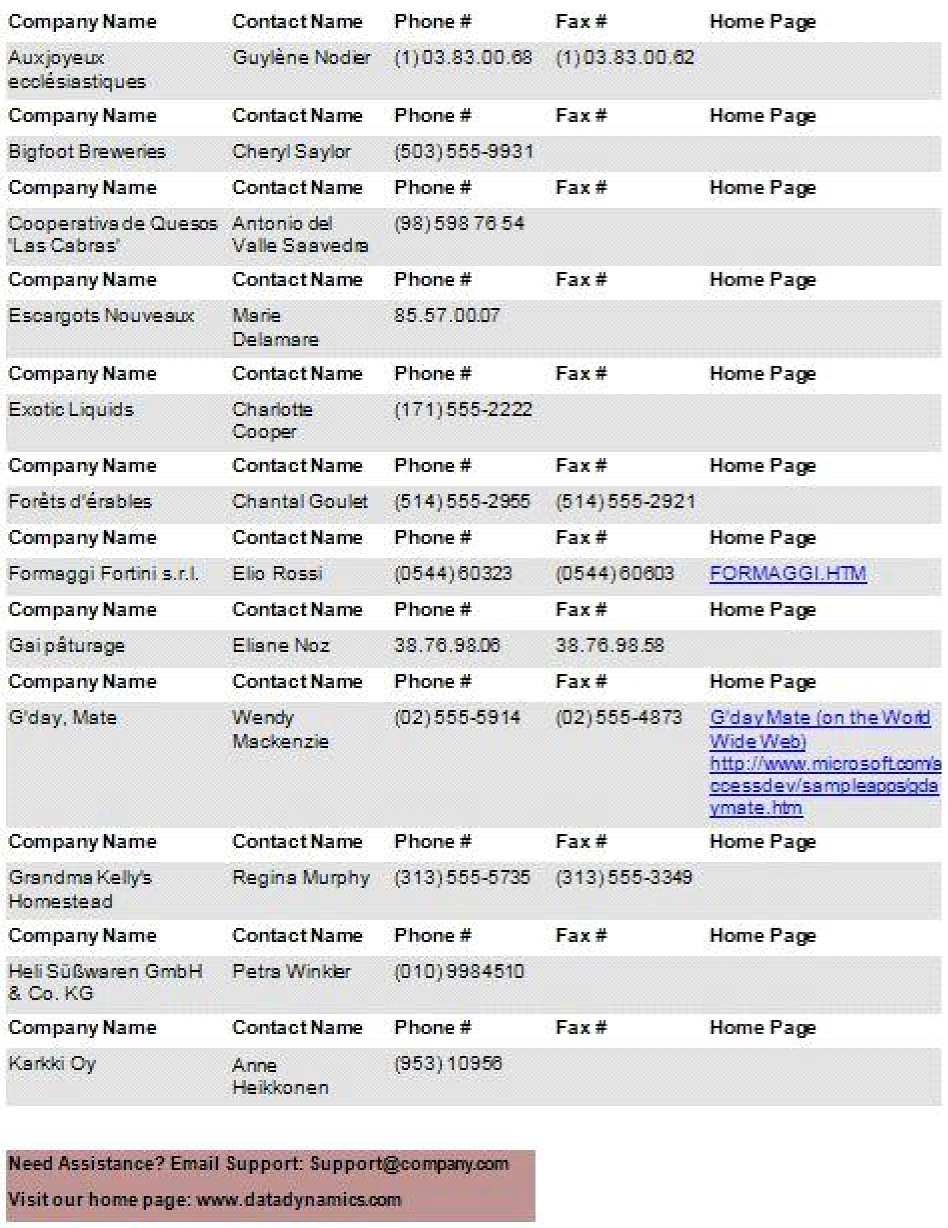
This walkthrough illustrates how to add to add hyperlinks to a report in the PageFooter section.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to display data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Company Name	Contact Name	Phone #	Fax #	Home Page
Auxjoyeux ecclésiastiques	Guylène Nodier	(1) 03.83.00.68	(1) 03.83.00.62	
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Bigfoot Breweries	Cheryl Saylor	(503) 555-9931		
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Cooperativa de Quesos 'Las Cabras'	Antonio del Valle Saavedra	(98) 598 76 54		
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Escargots Nouveaux	Marie Delamare	85.57.00.07		
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Exotic Liquids	Charlotte Cooper	(171) 555-2222		
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Forêts d'érables	Chantal Goulet	(514) 555-2955	(514) 555-2921	
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Formaggi Fortini s.r.l.	Elio Rossi	(0544) 60323	(0544) 60603	FORMAGGI.HTM
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Gai pâturage	Eliane Noz	38.76.98.06	38.76.98.58	
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
G'day, Mate	Wendy Mackenzie	(02) 555-5914	(02) 555-4873	G'day Mate (on the World Wide Web) http://www.microsoft.com/accessdev/sampleapps/gdaymate.htm
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Grandma Kelly's Homestead	Regina Murphy	(313) 555-5735	(313) 555-3349	
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Heli Süßwaren GmbH & Co. KG	Petra Winkler	(010) 9984510		
Company Name	**Contact Name**	**Phone #**	**Fax #**	**Home Page**
Karkki Oy	Anne Heikkonen	(953) 10956		

Need Assistance? Email Support: Support@company.com

Visit our home page: www.datadynamics.com

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptHyper.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from suppliers order by CompanyName".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptHyper.
2. Make the following changes to the group header:
   - Change the name to ghSuppliers
   - Change the DataField property to CompanyName
   - Change the GroupKeepTogether property to All
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblCompanyName**	Company Name	0, 0
Label	**lblContactName**	Contact Name	1.562, 0
Label	**lblPhone**	Phone #	2.687, 0
Label	**lblFax**	Fax #	3.812, 0
Label	**lblHomePage1**	Home Page	4.875, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	0, 0
TextBox	ContactName	**txtContactName**	Contact Name	1.562, 0
TextBox	Phone	**txtPhone**	Phone	2.687, 0
TextBox	Fax	**txtFax**	Fax	3.812, 0
TextBox	HomePage	**txtHomePage**	Home Page	4.854, 0

5. Add the following controls to the PageFooter section:

Control	Name	Text/Caption	Misc Details	Location
Label	**lblEmail**	Need Assistance? E-mail Support: Support@company.com	HyperLink = mailto:support@company.com	0, 0.125
Label	**lblHomePage**	Visit our home page: www.datadynamics.com	HyperLink = www.datadynamics.com	0, 0.375

Hyperlinks and Bookmarks

ActiveReports allows you to use the hyperlink property to reference back to the bookmarks collection.

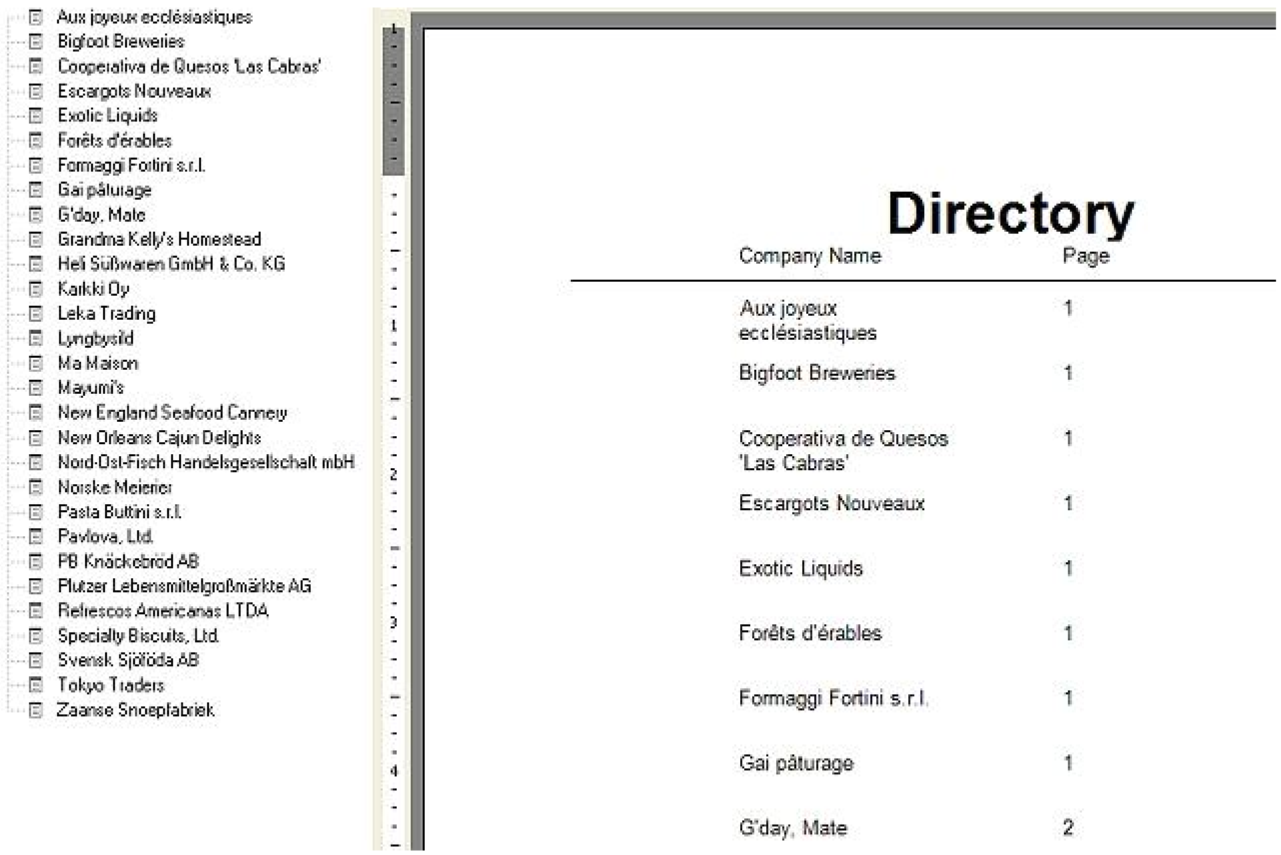
This walkthrough illustrates how to add to add hyperlinks to a report which reference items in the bookmarks collection and create a directory to match the items in the bookmarks collection.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the main report to a data source
- Adding controls to the reports to display data
- Adding code to the main report's Detail_Format event
- Adding code to the main report's ReportEnd event
- Adding code to the second report's Detail_Format event
- Adding code to the second report's FetchData event

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Aux joyeux ecclésiastiques
Bigfoot Breweries
Cooperativa de Quesos 'Las Cabras'
Escargots Nouveaux
Exotic Liquids
Forêts d'érables
Formaggi Fortini s.r.l.
Gai pâturage
G'day, Mate
Grandma Kelly's Homestead
Heli Süßwaren GmbH & Co. KG
Karkki Oy
Leka Trading
Lyngbysild
Ma Maison
Mayumi's
New England Seafood Cannery
New Orleans Cajun Delights
Nord-Ost-Fisch Handelsgesellschaft mbH
Norske Meierier
Pasta Buttini s.r.l.
Pavlova, Ltd.
PB Knäckebröd AB
Plutzer Lebensmittelgroßmärkte AG
Refrescos Americanas LTDA
Specialty Biscuits, Ltd.
Svensk Sjöföda AB
Tokyo Traders
Zaanse Snoepfabriek

Directory

Company Name	Page
Aux joyeux ecclésiastiques	1
Bigfoot Breweries	1
Cooperativa de Quesos 'Las Cabras'	1
Escargots Nouveaux	1
Exotic Liquids	1
Forêts d'érables	1
Formaggi Fortini s.r.l.	1
Gai pâturage	1
G'day, Mate	2

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMain.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptBookmarks.
7. Click **Open**.

Connecting rptMain to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**.
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from suppliers order by CompanyName".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptMain.
2. Make the following changes to the group header:
   - o Change the name to ghSuppliers
   - o Change the DataField property to CompanyName
   - o Change the GroupKeepTogether property to All
   - o Change the KeepTogether property to True
3. Add the following controls to **rptMain's** GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblCompanyName**	Company Name	0, 0
Label	**lblContactName**	Contact Name	1.562, 0
Label	**lblPhone**	Phone #	2.687, 0
Label	**lblFax**	Fax #	3.812, 0
Label	**lblHomePage1**	Home Page	4.875, 0

4. Add the following controls to **rptMain's** Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	0, 0
TextBox	ContactName	**txtContactName**	Contact Name	1.562, 0
TextBox	Phone	**txtPhone**	Phone	2.687, 0
TextBox	Fax	**txtFax**	Fax	3.812, 0
TextBox	HomePage	**txtHomePage**	Home Page	4.854, 0

5. Add the following controls to **rptMain's** PageFooter section:

Control	Name	Text/Caption	Misc Details	Location
Label	**lblEmail**	Need Assistance? E-mail Support: Support@company.com	HyperLink = mailto:support@company.com	0, 0.125
Label	**lblHomePage**	Visit our home page: www.datadynamics.com	HyperLink = www.datadynamics.com	0, 0.375

6. Add the following controls to **rptBookmarks'** PageHeader section:

Control	Name	Text/Caption	Location
Label	**lblDirectory**	Directory	0.5, 0
Label	**lblCompanyName**	CompanyName	1.125, 0.437
Label	**lblPage**	Page	3.312, 0.437

Line	**Line1**	(Empty string)	X1 = 0
			Y1 = 0.697
			X2 = 6.5
			Y2 = 0.697

7. Add the following controls to **rptBookmarks'** Detail section:

Control	Name	Text/Caption	Location
TextBox	**txtEntry**	Company Name	1.125, 0
TextBox	**txtPage**	Page	3.312, 0

Adding the code to rptMain's Detail_Format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptMain, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMain, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptMain's Detail_Format event.

To write the code in C#

- Click on the Detail section of rptMain to select the section. Click on the events icon in the **Properties** window to display available events for the report. Double-click *Format*. This creates an event-handling method for rptMain's Detail_Format event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e _
         As System.EventArgs) Handles Detail.Format
        Me.Detail.AddBookmark(Me.txtCompanyName.Text)
        Dim iStart As Integer
        Dim sHTML As String
        If txtHomePage.Text <> "" Then
            iStart = InStr(1, txtHomePage.Text, "#", _
                   CompareMethod.Text)
            sHTML = Right(txtHomePage.Text, (Len(txtHomePage.Text)_
                    - iStart))
            sHTML = Replace(sHTML, "#", "", 1, -1, CompareMethod.Text)
            txtHomePage.HyperLink = sHTML
            txtHomePage.Text = Replace(txtHomePage.Text, "#", "", _
                    1, -1, CompareMethod.Text)
        End If
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         this.Detail.AddBookmark(this.txtCompanyName.Text);
         int iStart;
         string sHTML;
         if (txtHomePage.Text != "")
         {
                  iStart = txtHomePage.Text.IndexOf("#",0);
                           //[1, txtHomePage.Text, "#", CompareMethod.Text);
                  sHTML = txtHomePage.Text.Substring(iStart,
                           txtHomePage.Text.Length - iStart);
                  sHTML = sHTML.Replace("#", "");
                  txtHomePage.HyperLink = sHTML;
                  txtHomePage.Text = txtHomePage.Text.Replace("#", "");
         }
}
```

Adding the code to rptMain's ReportEnd event

To write the code in Visual Basic

- Right-click in any section of the design surface of rptMain, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMain, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportEnd*. This creates an event-handling method for the rptMain's ReportEnd event.

To write the code in C#

- Click in the gray area below rptMain to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportEnd*. This creates an event-handling method for rptMain's ReportStart event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptMain_ReportEnd(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles MyBase.ReportEnd
        Dim rpt As New rptBookmarks()
        rpt.pBM = Me.Document.Bookmarks
        rpt.Run()
        Me.Document.Pages.InsertRange(Me.Document.Pages. _
                   Count, rpt.Document.Pages)
End Sub

[C#]

private void rptMain_ReportEnd(object sender, System.EventArgs eArgs)
{
         rptBookmarks rpt = new rptBookmarks();
         rpt.pBM = this.Document.Bookmarks;
         rpt.Run();
         this.Document.Pages.InsertRange(this.Document.Pages.Count,
                   rpt.Document.Pages);
}
```

Adding the code to rptBookmarks' Detail_Format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptBookmarks, and click on **View Code** to display the code view for the report. At the top left of the code view for rptBookmarks, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptBookmarks' Detail Format event.

To write the code in C#

- Click on the Detail section of rptBookmarks to select the section. Click on the events icon in the **Properties** window to display available events for the report. Double-click *Format*. This creates an event-handling method for rptBookmarks' Detail_Format event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Public pBM As New BookmarksCollection()
```

```
    Dim iEntry As Integer
    Private Sub Detail_Format(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Detail.Format
        Me.txtEntry.HyperLink = "toc://" + pBM(iEntry - 1).Label
        Me.txtEntry.Text = pBM(iEntry - 1).Label
        Me.txtPage.Text = pBM(iEntry - 1).PageNumber
End Sub

[C#]

public BookmarksCollection pBM = new BookmarksCollection();
private void Detail_Format(object sender, System.EventArgs eArgs)
{
         this.txtEntry.HyperLink = "toc://" + pBM[iEntry - 1].Label;
         this.txtEntry.Text = pBM[iEntry - 1].Label;
         this.txtPage.Text = pBM[iEntry - 1].PageNumber.ToString();
}
```

Adding code to rptBookmarks' FetchData event

To write the code in Visual Basic

- Right-click in any section of the design window of rptBookmarks, and click on **View Code** to display the code view for the report. At the top left of the code view for rptBookmarks, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptBookmarks' FetchData event.

To write the code in C#

- Click in the gray area below rptBookmarks to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptBookmarks' FetchData event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

  Private Sub rptBookmarks_FetchData(ByVal sender As Object, _
         ByVal eArgs As DataDynamics.ActiveReports. _
         ActiveReport.FetchEventArgs) Handles MyBase.FetchData
        If iEntry > pBM.Count - 1 Then
           eArgs.EOF = True
        Else
           eArgs.EOF = False
           iEntry += 1
        End If
End Sub

[C#]

int iEntry;
private void rptBookmarks_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
         if (iEntry > pBM.Count - 1)
         {
                  eArgs.EOF = true;
         }
         else
         {
                  eArgs.EOF = false;
                  iEntry += 1;
         }
}
```

Hyperlinks and Simulated Drill-Down Reporting

Hyperlinks can be used in Active Reports to simulate drill-down reporting.

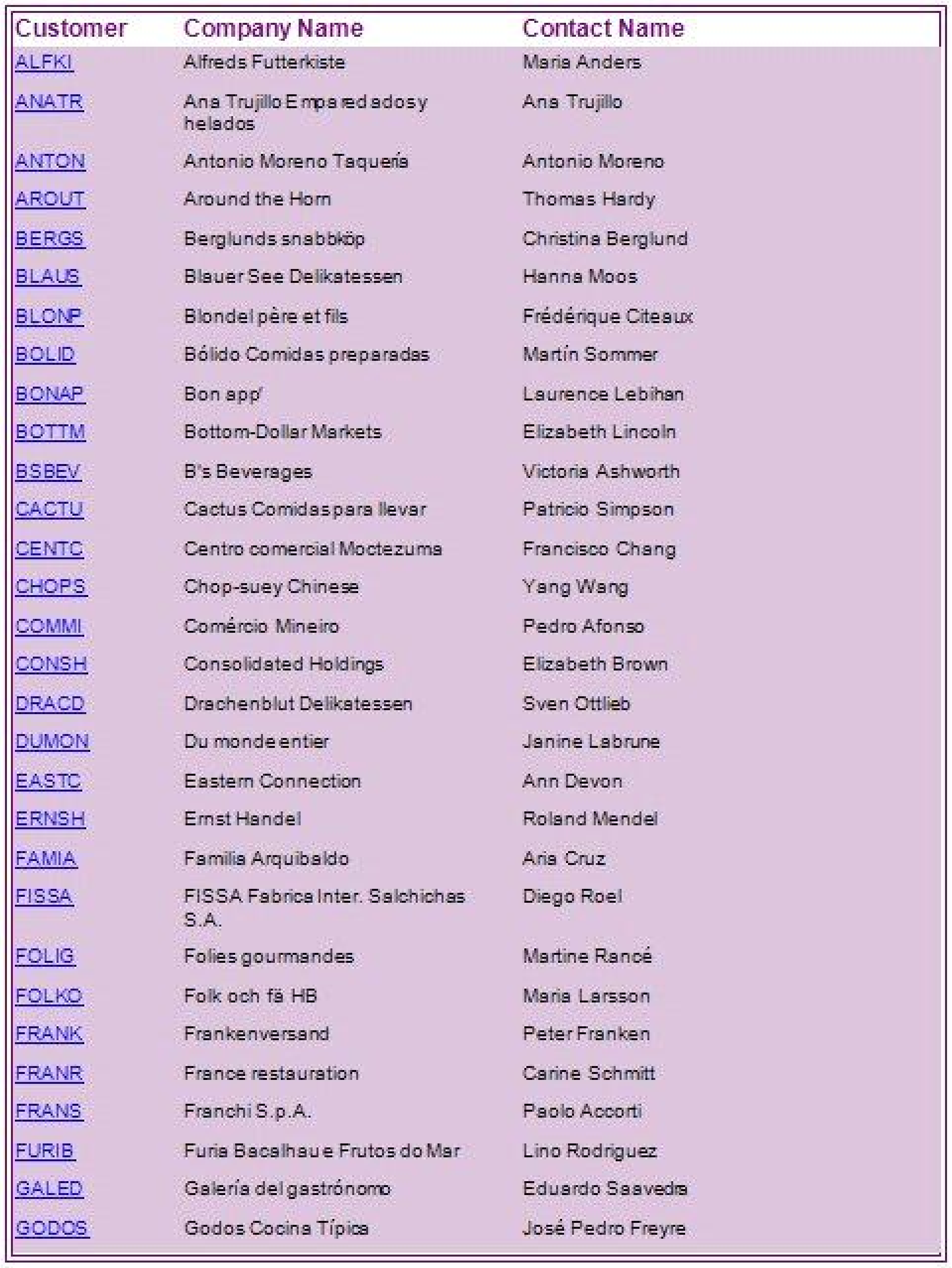
This walkthrough illustrates how to set up hyperlinks in a report to simulate drill-down reporting.

This walkthrough is split up into the following activities:

- Adding three ActiveReports to a Visual Studio project
- Connecting each report to a data source
- Adding controls to each report to display the data
- Adding three Windows Forms to the project
- Adding code to frmViewMain and frmViewDrillDown1
- Adding the code needed to set hyperlink properties

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Customer	Company Name	Contact Name
ALFKI	Alfreds Futterkiste	Maria Anders
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
ANTON	Antonio Moreno Taquería	Antonio Moreno
AROUT	Around the Horn	Thomas Hardy
BERGS	Berglunds snabbköp	Christina Berglund
BLAUS	Blauer See Delikatessen	Hanna Moos
BLONP	Blondel père et fils	Frédérique Citeaux
BOLID	Bólido Comidas preparadas	Martín Sommer
BONAP	Bon app'	Laurence Lebihan
BOTTM	Bottom-Dollar Markets	Elizabeth Lincoln
BSBEV	B's Beverages	Victoria Ashworth
CACTU	Cactus Comidas para llevar	Patricio Simpson
CENTC	Centro comercial Moctezuma	Francisco Chang
CHOPS	Chop-suey Chinese	Yang Wang
COMMI	Comércio Mineiro	Pedro Afonso
CONSH	Consolidated Holdings	Elizabeth Brown
DRACD	Drachenblut Delikatessen	Sven Ottlieb
DUMON	Du monde entier	Janine Labrune
EASTC	Eastern Connection	Ann Devon
ERNSH	Ernst Handel	Roland Mendel
FAMIA	Familia Arquibaldo	Aria Cruz
FISSA	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel
FOLIG	Folies gourmandes	Martine Rancé
FOLKO	Folk och fä HB	Maria Larsson
FRANK	Frankenversand	Peter Franken
FRANR	France restauration	Carine Schmitt
FRANS	Franchi S.p.A.	Paolo Accorti
FURIB	Furia Bacalhau e Frutos do Mar	Lino Rodriguez
GALED	Galería del gastrónomo	Eduardo Saavedra
GODOS	Godos Cocina Típica	José Pedro Freyre

Adding three ActiveReports to a Visual Studio project

To add three ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMain.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptDrillDown1.
7. Click **Open**.
8. Click on **Project > Add New Item**.

9. Select **ActiveReports file** and rename the file rptDrillDown2.
10. Click **Open**.

Connecting rptMain to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers order by customerID".
7. Click **OK** to return to the report design surface.

Connecting rptDrillDown1 to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from orders where customerID = '<%customerID%>' order by orderdate".
7. Click **OK** to return to the report design surface.

Connecting rptDrillDown2 to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from [order details] where orderID = <%orderID%> order by productid".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add the following controls to rptMain, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
Label	(Empty string)	**lblCustomer**	Customer	PageHeader	0, 0
Label	(Empty string)	**lblCompanyName**	Company Name	PageHeader	1.1875, 0
Label	(Empty string)	**lblContactName**	Contact Name	PageHeader	3.5625, 0
TextBox	CustomerID	**txtCustomerID**	Customer ID	Detail	0, 0
TextBox	CompanyName	**txtCompanyName**	Company Name	Detail	1.1875, 0
TextBox	ContactName	**txtContactName**	Contact Name	Detail	3.5625, 0

2. Add the following controls to rptDrillDown1, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
Label	(Empty string)	**lblCustomerID**	CustomerID	PageHeader	0, 0
Label	(Empty string)	**lblOrderID**	OrderID	PageHeader	1.1875, 0
Label	(Empty string)	**lblEmployeeID**	EmployeeID	PageHeader	2.4375, 0
Label	(Empty string)	**lblOrderDate**	Order Date	PageHeader	3.625, 0
Label	(Empty string)	**lblShippedDate**	Shipped Date	PageHeader	4.8125, 0
TextBox	CustomerID	**txtCustomerID**	Customer ID	Detail	0, 0
TextBox	OrderID	**txtOrderID**	Order ID	Detail	1.1875, 0
TextBox	EmployeeID	**txtEmployeeID**	Employee ID	Detail	2.4375, 0
TextBox	OrderDate	**txtOrderDate**	Order Date	Detail	3.625, 0
TextBox	ShippedDate	**txtShippedDate**	Shipped Date	Detail	4.8125, 0

3. Add the following controls to rptDrillDown2, naming them as indicated:

Control	DataField	Name	Text/Caption	Misc Details	Section	Location
Label	(Empty string)	**lblOrderID**	Order ID	(Empty string)	PageHeader	0, 0
Label	(Empty string)	**lblProductID**	Product ID	(Empty string)	PageHeader	1.1875, 0
Label	(Empty string)	**lblUnitPrice**	Unit Price	(Empty string)	PageHeader	2.375, 0
Label	(Empty string)	**lblQuantity**	Quantity	(Empty string)	PageHeader	3.5625, 0
Label	(Empty string)	**lblDiscount**	Discount	(Empty string)	PageHeader	4.75, 0
TextBox	OrderID	**txtOrderID**	Order ID	(Empty string)	Detail	0, 0
TextBox	ProductID	**txtProductID**	Product ID	(Empty string)	Detail	1.1875, 0
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	OutputFormat = Currency	Detail	2.375, 0
TextBox	Quantity	**txtQuantity**	Quantity	(Empty string)	Detail	3.5625, 0
TextBox	Discount	**txtDiscount**	Discount	OutputFormat = Currency	Detail	4.75, 0

Adding three Windows Forms to your project

To add three Windows Forms to your project

1. Click on **Project > Add Windows Form**.
2. Select **Windows Form** and rename it frmViewMain.
3. Click **Open**.
4. Click on **Project > Add Windows Form**.
5. Select **Windows Form** and rename it frmViewDrillDown1.
6. Click **Open**.
7. Click on **Project > Add Windows Form**.
8. Select **Windows Form** and rename it frmViewDrillDown2.
9. Click **Open**.

Adding code to frmViewMain

To write the code in Visual Basic

- Double-click at the top of frmViewMain to display the code view for the report. At the top left of the code view for frmViewMain, click the drop-down arrow and select *Viewer1*. At the top right of the code window, click the drop-down arrow and select *Hyperlink*. This creates an event-handling method for frmViewMain's Viewer1_Hyperlink event.

To write the code in C#

- Click in the Viewer section of frmViewMain to select the viewer. Click on the events icon in the **Properties** window to display available events for the viewer. Double-click *Hyperlink*. This creates an event-handling method for frmViewMain's Viewer1_Hyperlink event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Viewer1_HyperLink(ByVal sender As Object, ByVal _
          e As DataDynamics.ActiveReports.Viewer._
          HyperLinkEventArgs) Handles Viewer1.HyperLink
        Dim rpt2 As New rptDrillDown1()
        Dim frm2 As New frmViewDrillDown1()
        rpt2.Parameters("customerID").Value = e.HyperLink.ToString
        Console.WriteLine(rpt2.ds.SQL.ToString)
        rpt2.Run()
        frm2.Viewer1.Document = rpt2.Document
        frm2.ShowDialog(Me)
End Sub

[C#]

private void viewer1_HyperLink(object sender,
          DataDynamics.ActiveReports.Viewer.HyperLinkEventArgs e)
{
          rptDrillDown1 rpt2 = new rptDrillDown1();
          frmViewDrillDown1 frm2 = new frmViewDrillDown1();
          rpt2.Parameters["customerID"].Value = e.HyperLink.ToString();
          rpt2.Run();
          frm2.viewer1.Document = rpt2.Document;
          frm2.ShowDialog(this);
}
```

Adding code to frmViewDrillDown1

To write the code in Visual Basic

- Double-click at the top of frmViewDrillDown1 to display the code view for the report. At the top left of the code view for frmViewDrillDown1, click the drop-down arrow and select *Viewer1*. At the top right of the code window, click the drop-down arrow and select *Hyperlink*. This creates an event-handling method for frmViewDrillDown1's Viewer1_Hyperlink event.

To write the code in C#

- Click in the Viewer section of frmViewDrillDown1 to select the viewer. Click on the events icon in the **Properties** window to display available events for the viewer. Double-click *Hyperlink*. This creates an event-handling method for frmViewDrillDown's Viewer1_Hyperlink event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Viewer1_HyperLink(ByVal sender As Object, ByVal _
          e As DataDynamics.ActiveReports. _
          Viewer.HyperLinkEventArgs) Handles Viewer1.HyperLink
        Dim rpt3 As New rptDrillDown2()
        Dim frm3 As New frmViewDrillDown2()
        rpt3.Parameters("orderID").Value = e.HyperLink.ToString
        Console.WriteLine(rpt3.ds.SQL.ToString)
        rpt3.Run()
        frm3.Viewer1.Document = rpt3.Document
        frm3.ShowDialog(Me)
End Sub

[C#]

private void viewer1_HyperLink(object sender,
          DataDynamics.ActiveReports.Viewer.HyperLinkEventArgs e)
{
          rptDrillDown2 rpt3 = new rptDrillDown2();
          frmViewDrillDown2 frm3 = new frmViewDrillDown2();
          rpt3.Parameters["orderID"].Value = e.HyperLink.ToString();
          Console.WriteLine(rpt3.ds.SQL.ToString());
          rpt3.Run();
          frm3.viewer1.Document = rpt3.Document;
          frm3.ShowDialog(this);
}
```

Adding the code needed to set the hyperlink property for rptMain

To write the code in Visual Basic

- Right-click in any section of the design window of rptMain, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMain, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *BeforePrint*. This creates an event-handling method for rptMain's Detail_BeforePrint event.

To write the code in C#

- Click in the Detail section of rptMain to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *BeforePrint*. This creates an event-handling method for rptMain's Detail_BeforePrint event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_BeforePrint(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles Detail.BeforePrint
        Me.txtCustomerID.HyperLink = Me.txtCustomerID.Text
End Sub

[C#]

private void Detail_BeforePrint(object sender, System.EventArgs eArgs)
{
         this.txtCustomerID.HyperLink = this.txtCustomerID.Text;
}
```

Adding the code needed to set the hyperlink property for rptDrillDown1

To write the code in Visual Basic

- Right-click in any section of the design window of rptDrillDown1, and click on **View Code** to display the code view for the report. At the top left of the code view for rptDrillDown1, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *BeforePrint*. This creates an event-handling method for rptDrillDown1's Detail_BeforePrint event.

To write the code in C#

- Click in the Detail section of rptDrillDown1 to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *BeforePrint*. This creates an event-handling method for rptDrillDown1's Detail_BeforePrint event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_BeforePrint(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles Detail.BeforePrint
        Me.txtOrderID.HyperLink = Me.txtOrderID.Text
End Sub

[C#]

private void Detail_BeforePrint(object sender, System.EventArgs eArgs)
{
         this.txtOrderID.HyperLink = txtOrderID.Text;
}
```

Master Detail Reports Walkthroughs

With ActiveReports, Master Detail reports can be created quickly and easily. The following walkthroughs describe how to create different types of Master Detail reports.

- Master Detail Reports with Grouping
- Master Detail Reports with Subreports

- o Master Detail Reports with XML Data

Master Detail Reports with Grouping

ActiveReports allows you to create Master Detail reports with grouping by using the GroupHeader and Detail sections to contain data from master files and detail files.

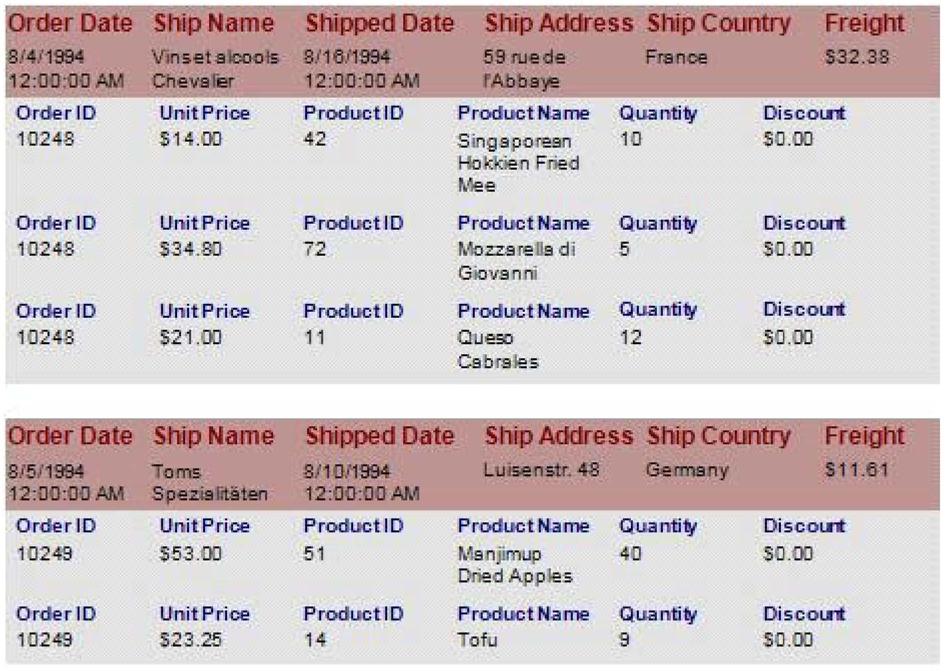
This walkthrough illustrates how to create a Master Detail report using grouping to organize the report.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Order Date	Ship Name	Shipped Date	Ship Address	Ship Country	Freight
8/4/1994 12:00:00 AM	Vins et alcools Chevalier	8/16/1994 12:00:00 AM	59 rue de l'Abbaye	France	$32.38

Order ID	Unit Price	Product ID	Product Name	Quantity	Discount
10248	$14.00	42	Singaporean Hokkien Fried Mee	10	$0.00
10248	$34.80	72	Mozzarella di Giovanni	5	$0.00
10248	$21.00	11	Queso Cabrales	12	$0.00

Order Date	Ship Name	Shipped Date	Ship Address	Ship Country	Freight
8/5/1994 12:00:00 AM	Toms Spezialitäten	8/10/1994 12:00:00 AM	Luisenstr. 48	Germany	$11.61

Order ID	Unit Price	Product ID	Product Name	Quantity	Discount
10249	$53.00	51	Manjimup Dried Apples	40	$0.00
10249	$23.25	14	Tofu	9	$0.00

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMD.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from orders, [order details], products where orders.OrderID = [order details].OrderID and products.productID = [order details].productID order by OrderDate".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to your report.
2. Make the following changes to the group header:
   - o Change the name to ghOrders
   - o Change the DataField property to Orders.OrderID
3. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Textbox	OrderDate	**txtOrderDate**	Order Date	(Empty string)	0, 0.25
Textbox	ShippedDate	**txtShippedDate**	Shipped Date	(Empty string)	2.062, 0.25
Label	(Empty string)	**lblOrderDate**	Order Date	(Empty string)	0, 0
Label	(Empty string)	**lblShipName**	Ship Name	(Empty string)	1, 0
Label	(Empty string)	**lblShippedDate**	Shipped Date	(Empty string)	2.062, 0
Label	(Empty string)	**lblFreight**	Freight	(Empty string)	5.687, 0
Textbox	Freight	**txtFreight**	Freight	OutputFormat = Currency	5.687, 0.25
Textbox	ShipName	**txtShipName**	Ship Name	(Empty string)	1, 0.25
Textbox	ShipAddress	**txtShipAddress**	Ship Address	(Empty string)	3.312, 0.25
Textbox	ShipCountry	**txtShipCountry**	Ship Country	(Empty string)	4.437, 0.25
Label	(Empty string)	**lblShipAddress**	Ship Address	(Empty string)	3.312, 0
Label	(Empty string)	**lblShipCountry**	Ship Country	(Empty string)	4.437, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	(Empty string)	**lblUnitPrice**	Unit Price	(Empty string)	1.062, 0

Label	(Empty string)	**lblQuantity**	Quantity	(Empty string)	4.25, 0
Label	(Empty string)	**lblDiscount**	Discount	(Empty string)	5.25, 0
Label	(Empty string)	**lblOrderID**	Order ID	(Empty string)	0.062, 0
Label	(Empty string)	**lblProductName**	Product Name	(Empty string)	3.125, 0
Textbox	ProductName	**txtProductName**	Product Name	(Empty string)	3.125, 0.187
Textbox	Quantity	**txtQuantity**	Quantity	(Empty string)	4.25, 0.1875
Textbox	Discount	**txtDiscount**	Discount	OutputFormat = Currency	5.25, 0.187
Textbox	orders.OrderID	**txtOrderID**	Order ID	(Empty string)	0.0625, 0.1875
Textbox	products.UnitPrice	**txtUnitPrice**	Unit Price	OutputFormat = Currency	1.0625, 0.1875
Textbox	products.ProductID	**txtProductID**	Product ID	(Empty string)	2.062, 0.187
Label	(Empty string)	**lblProductID**	Product ID	(Empty string)	2.062, 0

Master Detail Reports with Subreports

ActiveReports allows you to create Master Detail reports by using subreports to retrieve and group data.

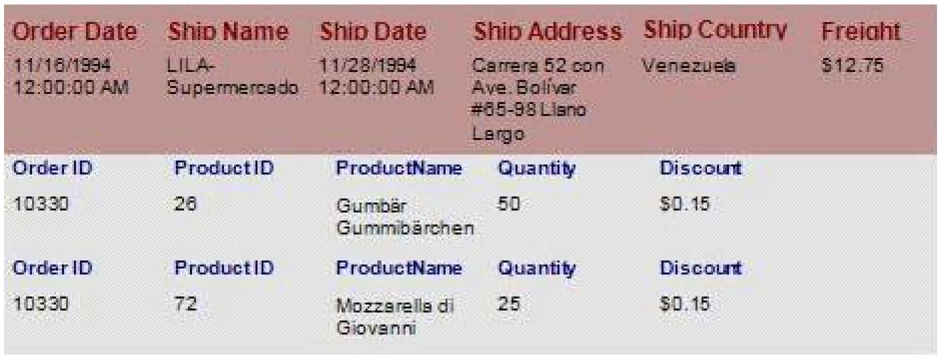
This walkthrough illustrates how to create a Master Detail report with subreports.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the Master report to a data source
- Connecting the Detail report to a data source
- Adding controls to the report to contain data
- Adding code to retrieve the subreport data at run time

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

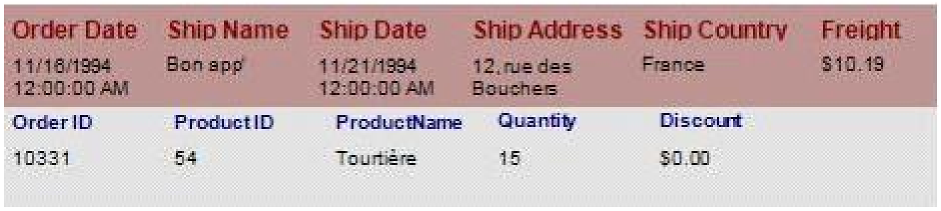
When you have completed this walkthrough, you will have a report that looks similar to the following.



Order Date	Ship Name	Ship Date	Ship Address	Ship Country	Freight
11/16/1994 12:00:00 AM	LILA-Supermercado	11/28/1994 12:00:00 AM	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Venezuela	$12.75

Order ID	Product ID	ProductName	Quantity	Discount
10330	26	Gumbär Gummibärchen	50	$0.15

Order ID	Product ID	ProductName	Quantity	Discount
10330	72	Mozzarella di Giovanni	25	$0.15



Order Date	Ship Name	Ship Date	Ship Address	Ship Country	Freight
11/16/1994 12:00:00 AM	Bon app'	11/21/1994 12:00:00 AM	12, rue des Bouchers	France	$10.19

Order ID	Product ID	ProductName	Quantity	Discount
10331	54	Tourtière	15	$0.00

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMaster.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptDetail.
7. Click **Open**.

Connecting the Master report to a data source

To connect the Master report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from orders".
7. Click **OK** to return to the report design surface.

Connecting the Detail report to a data source

To connect the Detail report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**

3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from [order details] inner join products on [order details].productid = products.productID where [order details].orderID = <%OrderID%>".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add the following controls to the Detail section of rptMaster:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	(Empty string)	**lblOrderDate**	Order Date	(Empty string)	0.0625, 0.0625
Label	(Empty string)	**lblShipName**	Ship Name	(Empty string)	1.125, 0.0625
Label	(Empty string)	**lblShipDate**	Shipped Date	(Empty string)	2.187, 0.0625
Label	(Empty string)	**lblShipAddress**	Ship Address	(Empty string)	3.25, 0.0625
Label	(Empty string)	**lblShipCountry**	Ship Country	(Empty string)	4.437, 0.0625
Label	(Empty string)	**lblFreight**	Freight	(Empty string)	5.687, 0.0625
Textbox	OrderDate	**txtOrderDate**	Order Date	(Empty string)	0.0625, 0.3125
Textbox	ShipName	**txtShipName**	Ship Name	(Empty string)	1.125, 0.3125
Textbox	ShippedDate	**txtShippedDate**	Shipped Date	(Empty string)	2.1875, 0.3125
Textbox	ShipAddress	**txtShipAddress**	Ship Address	(Empty string)	3.25, 0.3125
Textbox	ShipCountry	**txtShipCountry**	Ship Country	(Empty string)	4.4375, 0.3125
Textbox	Freight	**txtFreight**	Freight	OutputFormat = Currency	5.687, 0.3125
Subreport	(Empty string)	**ctlSubreport**	(Empty string)	(Empty string)	0, 0.5625
Label	(Empty string)	**lblWhiteLine**	(Empty string)	Background Color = White	0, 1.75

2. Add the following controls to the Detail section of rptDetail:

Control	DataField	Name	Text/Caption	Misc Details	Location
Textbox	OrderID	**txtOrderID**	Order ID	(Empty string)	0.0625, 0.25
Textbox	ProductName	**txtProductName**	Product Name	(Empty string)	2.3125, 0.25
Textbox	products.ProductID	**txtProductID**	Product ID	(Empty string)	1.1875, 0.25

Textbox	Quantity	**txtQuantity**	Quantity	(Empty string)	3.4375, 0.25
Textbox	Discount	**txtDiscount**	Discount	OutputFormat = Currency	4.5625, 0.25
Label	(Empty string)	**lblOrderID**	Order ID	(Empty string)	0.0625, 0
Label	(Empty string)	**lblProductID**	Product ID	(Empty string)	1.1875, 0
Label	(Empty string)	**lblProductName**	Product Name	(Empty string)	2.3125, 0
Label	(Empty string)	**lblQuantity**	Quantity	(Empty string)	3.4275, 0
Label	(Empty string)	**lblDiscount**	Discount	(Empty string)	4.5625, 0

Adding code to retrieve subreport data during run time

To write the code in Visual Basic

- Right-click in any section of the design window of rptMaster, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMaster, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptMaster's Detail_Format event. Add code to the handler to:
  - Retrieve subreport data at run time

To write the code in C#

- Click in the Detail section of rptMaster to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptMaster's Detail_Format event. Add code to the handler to:
  - Retrieve subreport data at run time

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal _
          e As System.EventArgs) Handles Detail.Format
        Dim rpt As New rptDetail()
        Me.ctlSubreport.Report = rpt

End Sub

[C#]

private void Detail_Format (object sender, System.EventArgs e)
{
          rptDetail rpt = new rptDetail();
          this.ctlSubReport.Report = rpt;
}
```

Master Detail Reports with XML Data

ActiveReports allows you to create Master Detail reports with data from a XML database.

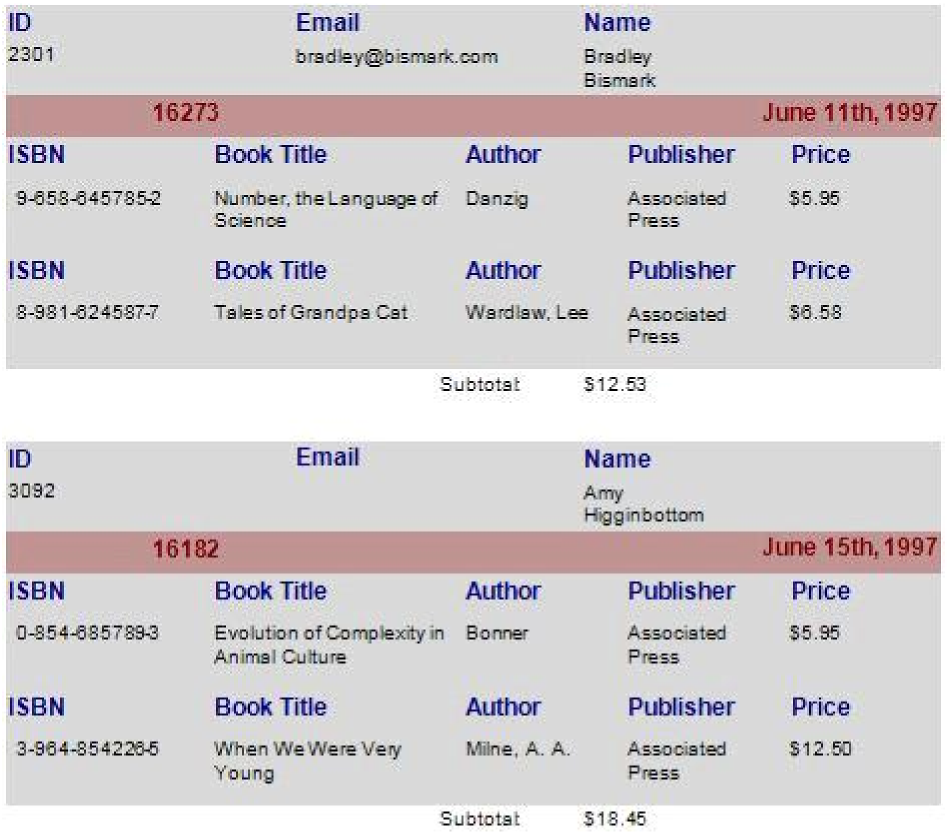
This walkthrough illustrates how to create a Master Detail report using XML data and grouping.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a XML data source
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the XML Customer database (Customer.xml).

When you have completed this walkthrough, you will have a report that looks similar to the following.



ID	Email	Name		
2301	bradley@bismark.com	Bradley Bismark		
16273				June 11th, 1997
ISBN	Book Title	Author	Publisher	Price
9-658-645785-2	Number, the Language of Science	Danzig	Associated Press	$5.95
ISBN	Book Title	Author	Publisher	Price
8-981-624587-7	Tales of Grandpa Cat	Wardlaw, Lee	Associated Press	$6.58
		Subtotal	$12.53	

ID	Email	Name		
3092		Amy Higginbottom		
16182				June 15th, 1997
ISBN	Book Title	Author	Publisher	Price
0-854-685789-3	Evolution of Complexity in Animal Culture	Bonner	Associated Press	$5.95
ISBN	Book Title	Author	Publisher	Price
3-964-854226-5	When We Were Very Young	Milne, A. A.	Associated Press	$12.50
		Subtotal	$18.45	

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMD.
4. Click **Open**.

Connecting the report to a XML data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on the Data Source drop-down arrow and select *XML*.
3. Click on the ellipsis beside *File URL* to browse for the access path to Customer.xml. Click **Open** once you have selected the appropriate access path.
4. In the Recordset Pattern field, type "//ITEM".
5. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add two GroupHeader/Footer sections to your report.
2. Make the following changes to the group header below the page header:
   - o Change the name to ghCustomer
   - o Change the DataField property to ../../@id
3. Make the following changes to the group header above the Detail section:
   - o Change the name to ghOrders
   - o Change the DataField property to ../Number
4. Add the following controls to ghCustomer:

Control	DataField	Name	Text/Caption	Location
Label	(Empty string)	**lblID**	ID	0, 0
Label	(Empty string)	**lblEmailAddress**	E-mail Address	2, 0
Label	(Empty string)	**lblName**	Name	4, 0
TextBox	../../@id	**txtID**	ID	0, 0.25
TextBox	../../@email	**txtEmail**	Email	2, 0.25
TextBox	../../@NAME	**txtName**	Name	4, 0.25

5. Add the following controls to ghOrders:

Control	DataField	Name	Text/Caption	Location
TextBox	../NUMBER	**txtNumber**	Number	1, 0
TextBox	../DATE	**txtDate**	Date	4.510, 0

6. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	(Empty string)	**lblISBN**	ISBN	(Empty string)	0, 0
Label	(Empty string)	**lblBookTitle**	Book Title	(Empty string)	1.437, 0
Label	(Empty string)	**lblPrice**	Price	(Empty string)	5.458, 0
Label	(Empty string)	**lblAuthor**	Author	(Empty string)	3.187, 0
Label	(Empty string)	**lblPublisher**	Publisher	(Empty string)	4.312, 0
TextBox	@isbn	**txtISBN**	ISBN	(Empty string)	0.0625, 0.312
TextBox	TITLE	**txtTitle**	Title	(Empty string)	1.437, 0.312
TextBox	AUTHOR	**txtAuthor**	Author	(Empty string)	3.187, 0.312

TextBox	PUBLISHER	**txtPublisher**	Publisher	(Empty string)	4.312, 0.312
TextBox	PRICE	**txtPrice**	Price	OutputFormat = Currency	5.437, 0.312

7. Add the following controls to GroupFooter2:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	(Empty string)	**lblSubtotal**	Subtotal	(Empty string)	3, 0
TextBox	PRICE	**txtSubtotal**	Subtotal	OutputFormat = Currency SummaryType = SubTotal SummaryGroup = ghOrders	4, 0

Modifying Report Documents Walkthroughs

ActiveReports allows report documents to be modified easily. The following walkthroughs illustrate how report documents can be changed based on user needs.

- Adding Pages
- Applying Page Templates
- Merging Reports

Adding Pages

ActiveReports allows you to add pages to your report in Visual Studio for previewing in the viewer control or printing. The document containing the inserted pages can also be saved to an RDF file or exported.

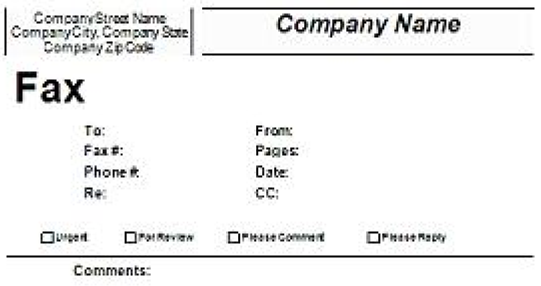
This walkthrough illustrates how to create two reports and insert the second report as a cover page for the first one.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the report to a data source
- Adding controls to both reports
- Adding code to the Form_Load event to insert the second report as a cover page

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



CompanyStreet Name
CompanyCity, Company State
Company Zip Code

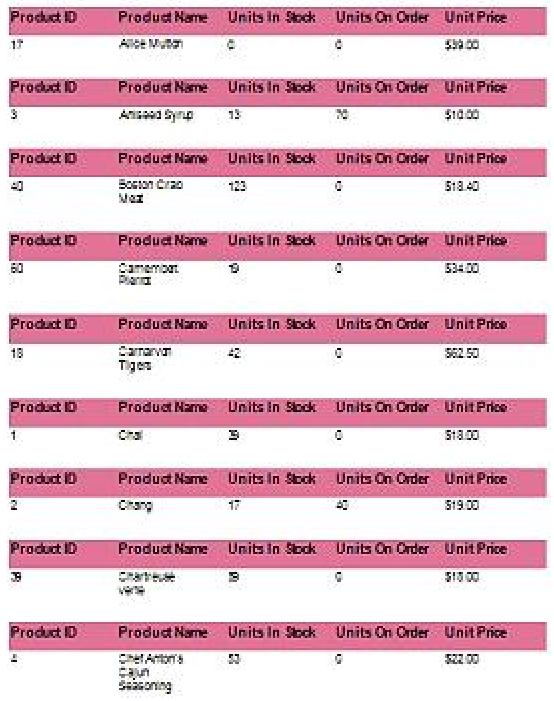
Company Name

Fax

To:	From:
Fax #:	Pages:
Phone #:	Date:
Re:	CC:

☐ Urgent ☐ For Review ☐ Please Comment ☐ Please Reply

Comments:



Product ID	Product Name	Units In Stock	Units On Order	Unit Price
17	Alice Mutton	0	0	$39.00
3	Aniseed Syrup	13	70	$10.00
40	Boston Crab Meat	123	0	$18.40
60	Camembert Pierrot	19	0	$34.00
18	Carnarvon Tigers	42	0	$62.50
1	Chai	39	0	$18.00
2	Chang	17	40	$19.00
39	Chartreuse verte	69	0	$18.00
4	Chef Anton's Cajun Seasoning	53	0	$22.00

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptInsertPage.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptCoverPage.
7. Click **Open**.

Connecting the report to a data source

To connect rptInsertPage to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products order by productname".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptInsertPage.
2. Make the following changes to the group header:
   - Change the name to ghProducts
   - Change the DataField property to ProductName
   - Change the GroupKeepTogether property to FirstDetail
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section of rptInsertPage:

Control	Name	Text/Caption	Location
Label	**lblProductID**	Product ID	0, 0
Label	**lblProductName**	Product Name	1.312, 0
Label	**lblUnitsInStock**	Units In Stock	2.625, 0
Label	**lblUnitsOnOrder**	Units On Order	3.937, 0
Label	**lblUnitPrice**	Unit Price	5.25, 0

4. Add the following controls to the Detail section of rptInsertPage:

Control	DataField	Name	Text/Caption	Misc Details	Location
TextBox	ProductID	**txtProductID**	Product ID	(Empty string)	0, 0
TextBox	ProductName	**txtProductName**	Product Name	(Empty string)	1.312, 0
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	(Empty string)	2.635, 0
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	OutputFormat = Currency	5.25, 0
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	(Empty string)	3.937, 0

5. Add the following controls to the PageHeader section of rptCoverPage:

Control	Name	Text/Caption	Section	Misc Details	Location
Label	**lblCompanyInfo**	Company Street Name Company City, Company State Company Zip Code	PageHeader	Text-Align = Center	0.0625, 0.0625
Label	**lblCompanyName**	Company Name	PageHeader	Text-Align = Center	2.437, 0.0625
Line	**Line1**	(Empty string)	PageHeader	(Empty string)	X1 = 2.437 Y1 = 0.0625 X2 = 6.437 Y2 = 0.0625
Line	**Line2**	(Empty string)	PageHeader	(Empty string)	X1 = 2.437 Y1 = 0.625 X2 = 6.437 Y2 = 0.625
Line	**Line3**	(Empty string)	PageHeader	(Empty string)	X1 = 0.0625 Y1 = 0.0625 X2 = 0.0625 Y2 = 0.625

Line	**Line4**	(Empty string)	PageHeader	(Empty string)	X1 = 2.312 Y1 = 0.0625 X2 = 2.312 Y2 = 0.625
TextBox	**txtComments**	Comments:	Detail	(Empty string)	0.875, 2.437

6. Add the following controls to the Detail section of rptCoverPage:

Control	Name	Text/Caption	Misc Details	Location
Label	**lblFax**	Fax	Text-Align = Center	0.0625, 0.0625
Label	**lblTo**	To:	(Empty string)	1, 0.75
Label	**lblFax2**	Fax #:	(Empty string)	1, 1
Label	**lblPhone**	Phone #:	(Empty string)	1, 1.25
Label	**lblRe**	Re:	(Empty string)	1, 1.5
TextBox	**txtTo**	(Empty string)	(Empty string)	1.875, 0.75
TextBox	**txtFax**	(Empty string)	(Empty string)	1.875, 1
TextBox	**txtPhone**	(Empty string)	(Empty string)	1.875, 1.25
TextBox	**txtRe**	(Empty string)	(Empty string)	1.875, 1.5
Label	**lblCC**	CC:	(Empty string)	3.0625, 1.5
Label	**lblDate**	Date:	(Empty string)	3.0625, 1.25
Label	**lblPages**	Pages:	(Empty string)	3.0625, 1
Label	**lblFrom**	From	(Empty string)	3.0625, 0.75
TextBox	**txtFrom**	(Empty string)	(Empty string)	3.937, 0.75
TextBox	**txtPages**	(Empty string)	(Empty string)	3.937, 1
TextBox	**txtDate**	(Empty string)	(Empty string)	3.937, 1.25
TextBox	**txtCC**	(Empty string)	(Empty string)	3.937, 1.5
CheckBox	**chkUrgent**	Urgent	(Empty string)	0.5, 2.0625
CheckBox	**chkForReview**	For Review	(Empty string)	1.5, 2.0625
CheckBox	**chkPleaseComment**	Please Comment	(Empty string)	2.75, 2.0625
CheckBox	**chkPleaseReply**	Please Reply	(Empty string)	4.437, 2.0625

Line	**Line5**	(Empty string)	(Empty string)	X1 = 0.0625
				Y1 = 2.375
				X2 = 6.437
				Y2 = 2.375

TextBox	**txtComments**	Comments:	(Empty string)	0.875, 2.437

Adding code to the Form_Load event

To write the code in Visual Basic

- Right-click at the top of Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
    - Insert rptCoverPage at the beginning of rptInsertPage

To write the code in C#

- Click at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for the form. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

  - Insert rptCoverPage at the beginning of rptInsertPage

The following example shows what the code for the Insert method looks like:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, ByVal _
          e As System.EventArgs) Handles MyBase.Load
          Dim rpt As New rptInsertPage()
          Viewer1.Document = rpt.Document
          rpt.Run()
          Dim rpt2 As New rptCoverPage()
          rpt2.Run()
          rpt.Document.Pages.Insert(0, rpt2.Document.Pages(0))
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
          rptInsertPage rpt = new rptInsertPage();
          rptCoverPage rpt2 = new rptCoverPage();
          viewer1.Document = rpt.Document;
          rpt.Run();
          rpt2.Run();
          rpt.Document.Pages.Insert(0, rpt2.Document.Pages[0]);
}
```

The following example shows what the code for the InsertNew method looks like. This method allows you to insert a new page at the location you specify:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, ByVal _
          e As System.EventArgs) Handles MyBase.Load
          Dim rpt As New rptInsertPage()
          Viewer1.Document = rpt.Document
          rpt.Run()
          rpt.Document.Pages.InsertNew(3)
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
          rptInsertPage rpt = new rptInsertPage();
          viewer1.Document = rpt.Document;
          rpt.Run();
          rpt.Document.Pages.InsertNew(3);
}
```

Applying Page Templates

ActiveReports allows you to apply a page template to an existing report by using the Overlay method.

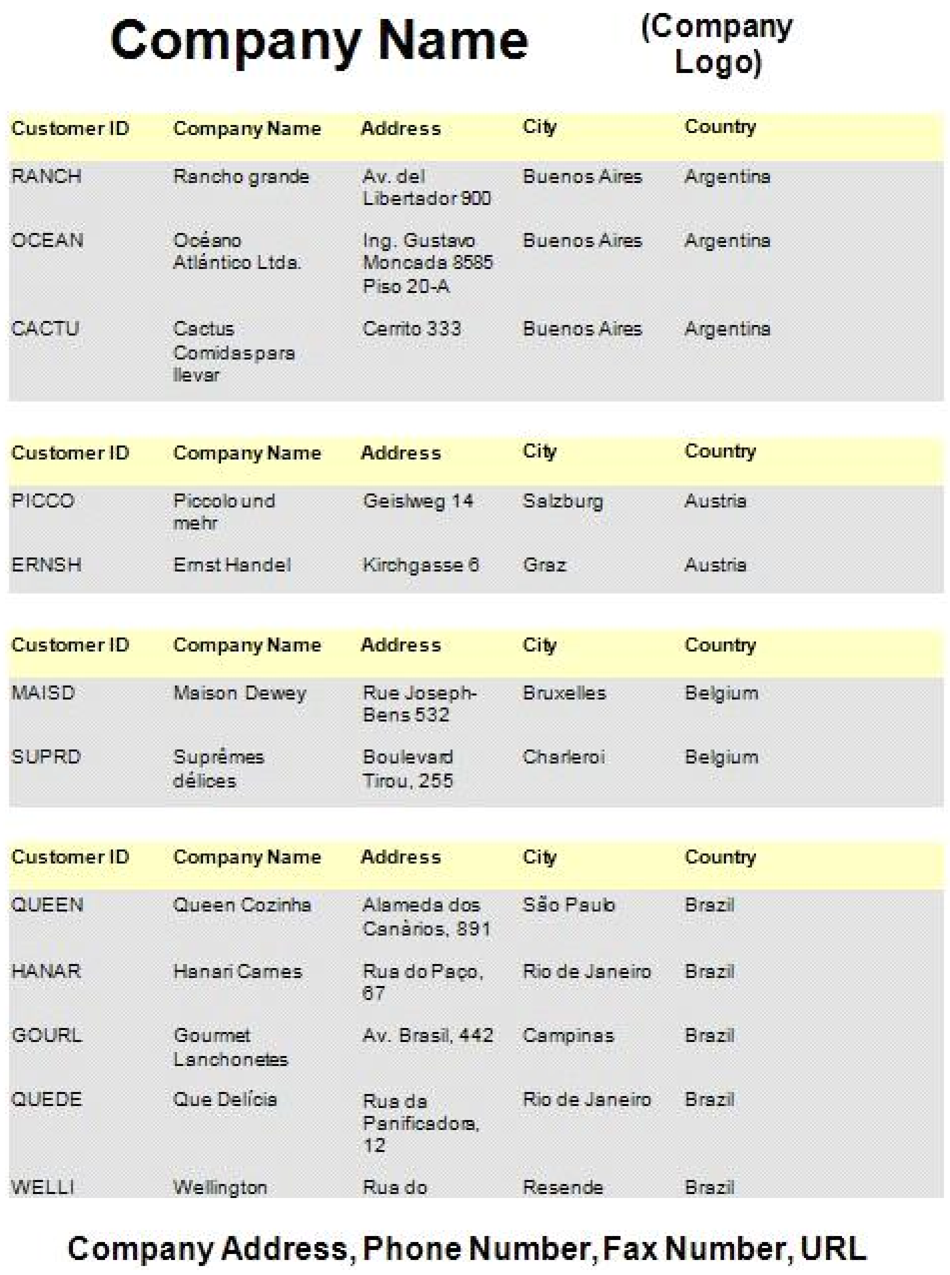
This walkthrough illustrates how to overlay an ActiveReport with a "letterhead" page template.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the main report to a data source
- Adding controls to both reports
- Adding code to the Form_Load event to overlay the report pages with the "letterhead" template

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Company Name **(Company Logo)**

Customer ID	Company Name	Address	City	Country
RANCH	Rancho grande	Av. del Libertador 900	Buenos Aires	Argentina
OCEAN	Océano Atlántico Ltda.	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	Argentina
CACTU	Cactus Comidas para llevar	Cerrito 333	Buenos Aires	Argentina

Customer ID	Company Name	Address	City	Country
PICCO	Piccolo und mehr	Geislweg 14	Salzburg	Austria
ERNSH	Ernst Handel	Kirchgasse 6	Graz	Austria

Customer ID	Company Name	Address	City	Country
MAISD	Maison Dewey	Rue Joseph-Bens 532	Bruxelles	Belgium
SUPRD	Suprêmes délices	Boulevard Tirou, 255	Charleroi	Belgium

Customer ID	Company Name	Address	City	Country
QUEEN	Queen Cozinha	Alameda dos Canàrios, 891	São Paulo	Brazil
HANAR	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	Brazil
GOURL	Gourmet Lanchonetes	Av. Brasil, 442	Campinas	Brazil
QUEDE	Que Delícia	Rua da Panificadora, 12	Rio de Janeiro	Brazil
WELLI	Wellington	Rua do	Resende	Brazil

Company Address, Phone Number, Fax Number, URL

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptReport.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptTemplate.
7. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers order by country".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptReport.
2. Make the following changes to the group header:
   - o Change the name to ghCustomers
   - o Change the DataField property to Country
3. Add the following controls to rptReport, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
Label	(Empty string)	**lblCustomerID**	Customer ID	GroupHeader	0, 0
Label	(Empty string)	**lblCompanyName**	Company Name	GroupHeader	1.125, 0
Label	(Empty string)	**lblAddress**	Address	GroupHeader	2.625, 0
Label	(Empty string)	**lblCity**	City	GroupHeader	4.125, 0
Label	(Empty string)	**lblCountry**	Country	GroupHeader	5.187, 0
TextBox	CustomerID	**txtCustomerID**	Customer ID	Detail	0, 0
TextBox	CompanyName	**txtCompanyName**	CompanyName	Detail	1.125, 0
TextBox	Address	**txtAddress**	Address	Detail	2.625, 0
TextBox	City	**txtCity**	City	Detail	4.125, 0
TextBox	Country	**txtCountry**	Country	Detail	5.187, 0

4. Add the following controls to rptTemplate, naming them as indicated:

Control	Name	Text/Caption	Section	Location
TextBox	**txtCompanyName**	Company Name	PageHeader	0.0625, 0.0625
Label	**lblCompanyLogo**	(Company Logo)	PageHeader	4, 0.0625
Label	**lblCompanyInfo**	Company Address, Phone Number, Fax Number, URL	PageFooter	0.0625, 0.0625

Adding code to the Form_Load event

To write the code in Visual Basic

- Right-click at the top of Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
  - Apply the "letterhead" template page to rptReport

To write the code in C#

- Click at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for the form. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
  - Apply the "letterhead" template page to rptReport

The following example shows what the code for the Overlay method looks like:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, ByVal _
         e As System.EventArgs) Handles MyBase.Load
         Dim rpt As New rptReport()
         rpt.Run()
         Dim rpt2 As New rptTemplate()
         rpt2.Run()
         Viewer1.Document = rpt.Document
         For i = 0 To rpt.Document.Pages.Count - 1
                  rpt.Document.Pages(i).Overlay(rpt2.Document.Pages(0))
         Next
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         rptReport rpt = new rptReport();
         rpt.Run();
         rptTemplate rpt2 = new rptTemplate();
         rpt2.Run();
         viewer1.Document = rpt.Document;
         for(int i = 0; i < rpt.Document.Pages.Count; i++)
         {
                  rpt.Document.Pages[i].Overlay(rpt2.Document.Pages[0]);
         }
}
```

Merging Reports

ActiveReports allows you to merge two or more reports in Visual Studio for previewing in the viewer control or printing. The document containing the merged reports can also be saved to an RDF file or exported.

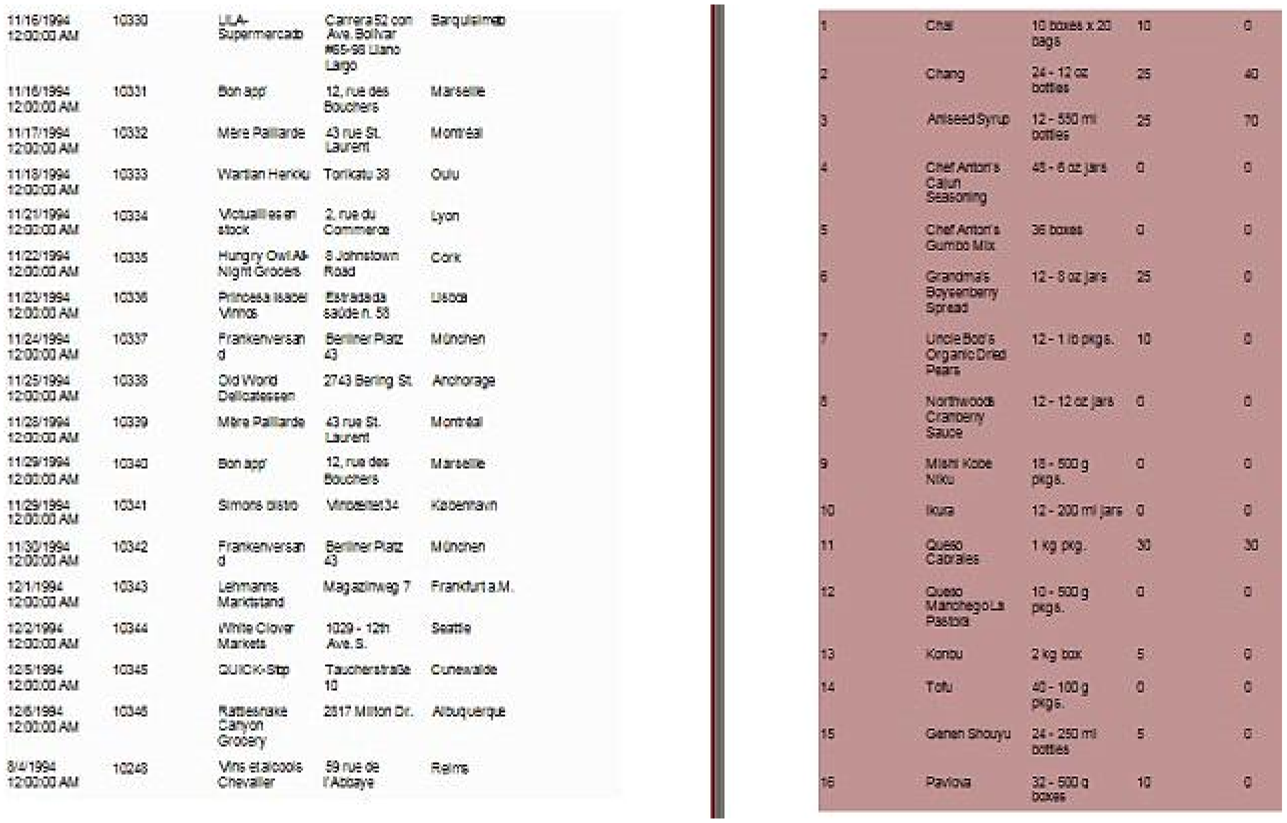
This walkthrough illustrates how to create two ActiveReports and merge the reports into one document.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the reports to a data source
- Adding controls to both reports to contain data
- Adding code to the Form_Load event to combine the reports

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



11/16/1994 12:00:00 AM	10330	LILA-Supermercado	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto
11/16/1994 12:00:00 AM	10331	Bon app'	12, rue des Bouchers	Marseille
11/17/1994 12:00:00 AM	10332	Mère Paillarde	43 rue St. Laurent	Montréal
11/18/1994 12:00:00 AM	10333	Wartian Herkku	Torikatu 38	Oulu
11/21/1994 12:00:00 AM	10334	Victuailles en stock	2, rue du Commerce	Lyon
11/22/1994 12:00:00 AM	10335	Hungry Owl All-Night Grocers	8 Johnstown Road	Cork
11/23/1994 12:00:00 AM	10336	Princesa Isabel Vinhos	Estrada da saúde n. 58	Lisboa
11/24/1994 12:00:00 AM	10337	Frankenversand	Berliner Platz 43	München
11/25/1994 12:00:00 AM	10338	Old World Delicatessen	2743 Bering St.	Anchorage
11/28/1994 12:00:00 AM	10339	Mère Paillarde	43 rue St. Laurent	Montréal
11/29/1994 12:00:00 AM	10340	Bon app'	12, rue des Bouchers	Marseille
11/29/1994 12:00:00 AM	10341	Simons bistro	Vinbæltet 34	København
11/30/1994 12:00:00 AM	10342	Frankenversand	Berliner Platz 43	München
12/1/1994 12:00:00 AM	10343	Lehmanns Marktstand	Magazinweg 7	Frankfurt a.M.
12/2/1994 12:00:00 AM	10344	White Clover Markets	1029 - 12th Ave. S.	Seattle
12/5/1994 12:00:00 AM	10345	QUICK-Stop	Taucherstraße 10	Cunewalde
12/6/1994 12:00:00 AM	10346	Rattlesnake Canyon Grocery	2817 Milton Dr.	Albuquerque
8/4/1994 12:00:00 AM	10248	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims

1	Chai	10 boxes x 20 bags	10	0
2	Chang	24 - 12 oz bottles	25	40
3	Aniseed Syrup	12 - 550 ml bottles	25	70
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	0	0
5	Chef Anton's Gumbo Mix	36 boxes	0	0
6	Grandma's Boysenberry Spread	12 - 8 oz jars	25	0
7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	10	0
8	Northwoods Cranberry Sauce	12 - 12 oz jars	0	0
9	Mishi Kobe Niku	18 - 500 g pkgs.	0	0
10	Ikura	12 - 200 ml jars	0	0
11	Queso Cabrales	1 kg pkg.	30	30
12	Queso Manchego La Pastora	10 - 500 g pkgs.	0	0
13	Konbu	2 kg box	5	0
14	Tofu	40 - 100 g pkgs.	0	0
15	Genen Shouyu	24 - 250 ml bottles	5	0
16	Pavlova	32 - 500 g boxes	10	0

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptOne.
4. Click **Open**.

5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptTwo.
7. Click **Open**.

Connecting the reports to a data source

To connect rptOne to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from orders".
7. Click **OK** to return to the report design surface.

To connect rptTwo to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add the following controls to rptOne, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	OrderDate	**txtOrderDate**	Order Date	0, 0
TextBox	OrderID	**txtOrderID**	Order ID	1.125, 0
TextBox	ShipName	**txtShipName**	Ship Name	2.25, 0
TextBox	ShipAddress	**txtShipAddress**	Ship Address	3.375, 0
TextBox	ShipCity	**txtShipCity**	Ship City	4.5, 0

2. Add the following controls to rptTwo, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductID	**txtProductID**	Product ID	0, 0.06
TextBox	ProductName	**txtProductName**	Product Name	1.125, 0.06
TextBox	QuantityPerUnit	**txtQuantityPerUnit**	Quantity Per Unit	2.25, 0.06
TextBox	ReorderLevel	**txtReorderLevel**	Reorder Level	3.375, 0.06
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	4.5, 0.06

Adding code to the Form_Load event

To write the code in Visual Basic

- Right-click at the top of Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
  - Add rptTwo to rptOne

To write the code in C#

- Click at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for the form. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
  - Add rptTwo to rptOne

The following example shows what the code for the AddRange method looks like:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, ByVal _
          e As System.EventArgs) Handles MyBase.Load
        Dim rpt As New rptOne()
        rpt.Run()
        Dim rpt2 As New rptTwo()
        rpt2.Run()
        rpt.Document.Pages.AddRange(rpt2.Document.Pages)
        Viewer1.Document = rpt.Document
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         rptOne rpt1 = new rptOne();
         rpt1.Run();
         rptTwo rpt2 = new rptTwo();
         rpt2.Run();
         rpt1.Document.Pages.AddRange(rpt2.Document.Pages);
         viewer1.Document = rpt1.Document;
}
```

The following example shows what the code for the Add method looks like:

```
[Visual Basic]

Dim i As Integer
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
         Dim rpt As New rptOne()
         rpt.Run()
         Dim rpt2 As New rptTwo()
         rpt2.Run()
         For i = 0 To rpt2.Document.Pages.Count - 1
                  rpt.Document.Pages.Add(rpt2.Document.Pages(i))
         Next
         Viewer1.Document = rpt.Document
End Sub
```

```
[C#]

int i;
private void Form1_Load(object sender, System.EventArgs e)
{
          rptOne rpt1 = new rptOne();
          rpt1.Run();
          rptTwo rpt2 = new rptTwo();
          rpt2.Run();
          for(i = 0; i < rpt2.Document.Pages.Count; i++)
          {
                    rpt1.Document.Pages.Add(rpt2.Document.Pages[i]);
          }
          viewer1.Document = rpt1.Document;
}
```

Page Numbering Walkthroughs

With ActiveReports, page numbering can be set up using either the report's PageHeader/Footer sections or GroupHeader/Footer section. The following walkthroughs illustrate how to set up page numbering in the GroupHeader section and in the PageFooter section.

- Page Numbering in the Group Header
- Page Numbering in the Page Footer

Page Numbering in the Group Header

With ActiveReports, page numbering can be easily applied to groups in a report using the GroupHeader section.

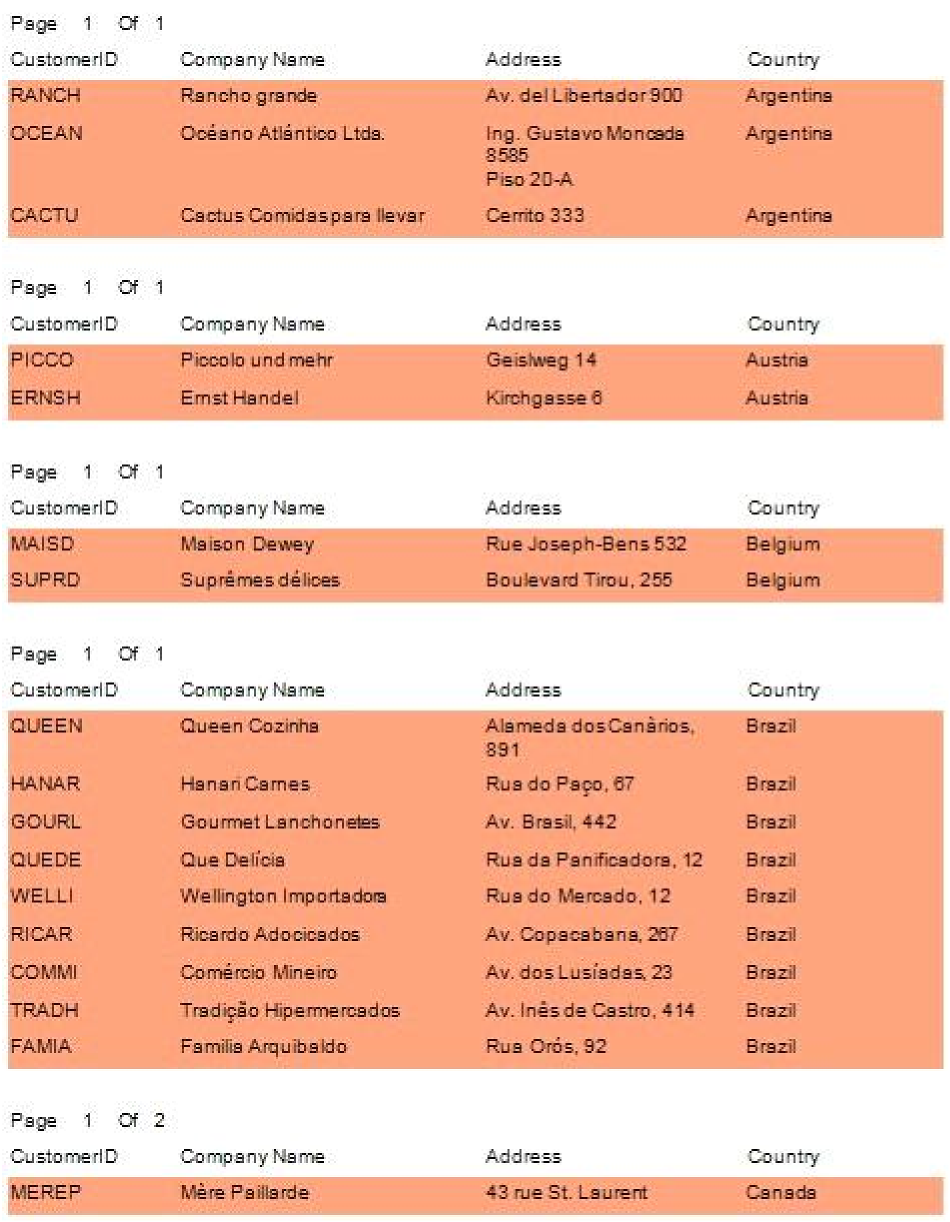
This walkthrough illustrates the basics of setting up page numbering for groups in the GroupHeader section.

The walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the data source to a database
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Page 1 Of 1

CustomerID	Company Name	Address	Country
RANCH	Rancho grande	Av. del Libertador 900	Argentina
OCEAN	Océano Atlántico Ltda.	Ing. Gustavo Moncada 8585 Piso 20-A	Argentina
CACTU	Cactus Comidas para llevar	Cerrito 333	Argentina

Page 1 Of 1

CustomerID	Company Name	Address	Country
PICCO	Piccolo und mehr	Geislweg 14	Austria
ERNSH	Ernst Handel	Kirchgasse 6	Austria

Page 1 Of 1

CustomerID	Company Name	Address	Country
MAISD	Maison Dewey	Rue Joseph-Bens 532	Belgium
SUPRD	Suprêmes délices	Boulevard Tirou, 255	Belgium

Page 1 Of 1

CustomerID	Company Name	Address	Country
QUEEN	Queen Cozinha	Alameda dos Canàrios, 891	Brazil
HANAR	Hanari Carnes	Rua do Paço, 67	Brazil
GOURL	Gourmet Lanchonetes	Av. Brasil, 442	Brazil
QUEDE	Que Delícia	Rua da Panificadora, 12	Brazil
WELLI	Wellington Importadora	Rua do Mercado, 12	Brazil
RICAR	Ricardo Adocicados	Av. Copacabana, 267	Brazil
COMMI	Comércio Mineiro	Av. dos Lusíadas, 23	Brazil
TRADH	Tradição Hipermercados	Av. Inês de Castro, 414	Brazil
FAMIA	Familia Arquibaldo	Rua Orós, 92	Brazil

Page 1 Of 2

CustomerID	Company Name	Address	Country
MEREP	Mère Paillarde	43 rue St. Laurent	Canada

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Click on **Project > Add New Item...**
2. Select **ActiveReports File** and rename it rptNumberGH.
3. Change the name of the report and click **Open**.
4. The ActiveReports design surface is displayed.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers order by country".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to contain data

1. Add a GroupHeader/Footer section to rptNumberGH.
2. Make the following changes to the group header:
   - o Change the name to ghCustomers
   - o Change the DataField property to Country
   - o Change the GroupKeepTogether property to FirstDetail
   - o Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Misc Details	Location
Label	**lblPage**	Page	(Empty string)	0, 0
Label	**lblOf**	Of	(Empty string)	0.75, 0
TextBox	**txtPageNumber**	#	SummaryType = PageCount SummaryRunning = Group SummaryGroup = ghCustomers	0.5, 0
TextBox	**txtPageCount**	##	SummaryType = PageCount SummaryGroup = ghCustomers	1, 0
Label	**lblCustomerID**	CustomerID	(Empty string)	0, 0.25
Label	**lblCompanyName**	CompanyName	(Empty string)	1.1875, 0.25
Label	**lblAddress**	Address	(Empty string)	3.3125, 0.25
Label	**lblCountry**	Country	(Empty string)	5.125, 0.25

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	CustomerID	**txtCustomerID**	CustomerID	0, 0
TextBox	CompanyName	**txtCompanyName**	CompanyName	1.1875, 0
TextBox	Address	**txtAddress**	Address	3.3125, 0
TextBox	Country	**txtCountry**	Country	5.114, 0

Page Numbering in the Page Footer

With ActiveReports, page numbering can be easily applied to a report using the PageFooter section.

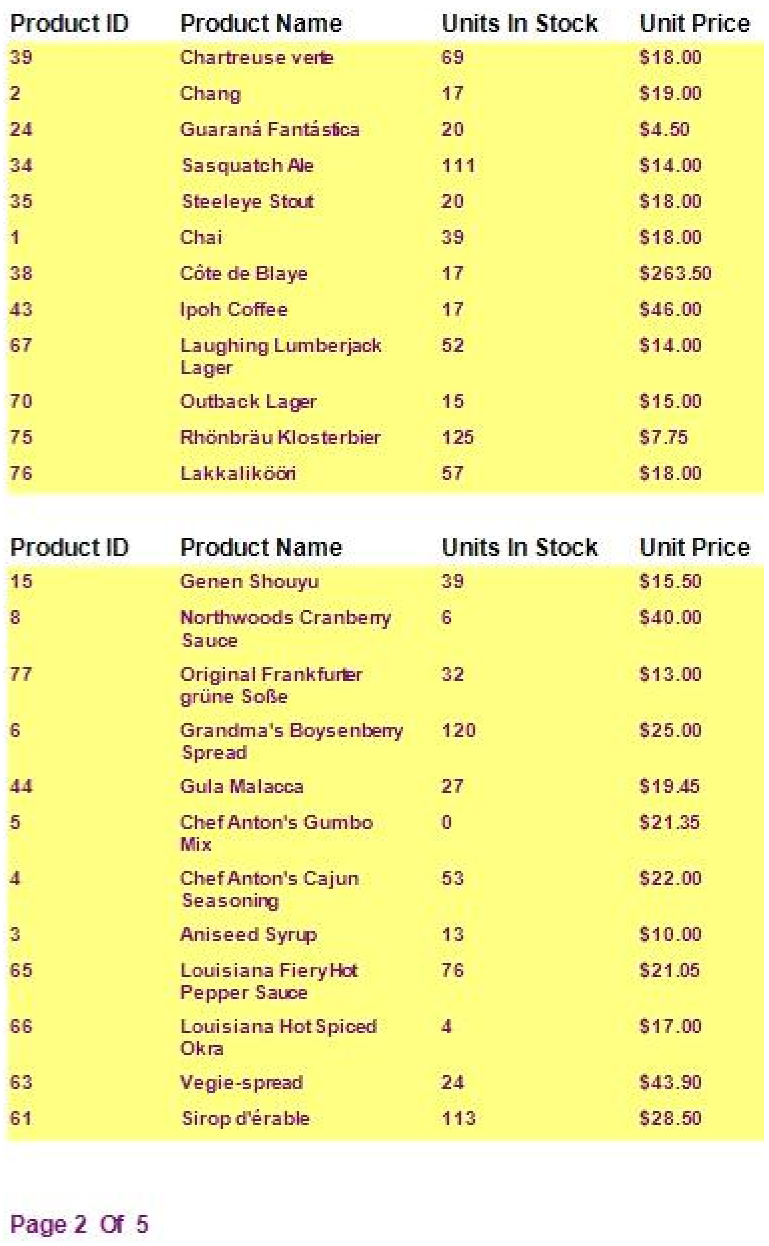
This walkthrough illustrates the basics of setting up page numbering in the PageFooter section.

The walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the data source to a database
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Product ID	Product Name	Units In Stock	Unit Price
39	Chartreuse verte	69	$18.00
2	Chang	17	$19.00
24	Guaraná Fantástica	20	$4.50
34	Sasquatch Ale	111	$14.00
35	Steeleye Stout	20	$18.00
1	Chai	39	$18.00
38	Côte de Blaye	17	$263.50
43	Ipoh Coffee	17	$46.00
67	Laughing Lumberjack Lager	52	$14.00
70	Outback Lager	15	$15.00
75	Rhönbräu Klosterbier	125	$7.75
76	Lakkalikööri	57	$18.00

Product ID	Product Name	Units In Stock	Unit Price
15	Genen Shouyu	39	$15.50
8	Northwoods Cranberry Sauce	6	$40.00
77	Original Frankfurter grüne Soße	32	$13.00
6	Grandma's Boysenberry Spread	120	$25.00
44	Gula Malacca	27	$19.45
5	Chef Anton's Gumbo Mix	0	$21.35
4	Chef Anton's Cajun Seasoning	53	$22.00
3	Aniseed Syrup	13	$10.00
65	Louisiana Fiery Hot Pepper Sauce	76	$21.05
66	Louisiana Hot Spiced Okra	4	$17.00
63	Vegie-spread	24	$43.90
61	Sirop d'érable	113	$28.50

Page 2 Of 5

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Click on **Project > Add New Item...**
2. Select **ActiveReports File** and rename it rptNumberPF.
3. Change the name of the report and click **Open**.
4. The ActiveReports design surface is displayed.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products order by categoryID".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to contain data

1. Add a GroupHeader/Footer section to rptNumberPF.
2. Make the following changes to the group header:
   - Change the name to ghProducts
   - Change the DataField property to categoryID
   - Change the GroupKeepTogether property to All
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductID**	Product ID	0, 0
Label	**lblProductName**	Product Name	1.1875, 0
Label	**lblUnitsInStock**	Units In Stock	3, 0
Label	**lblUnitPrice**	Unit Price	4.375, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Misc Details	Location
TextBox	ProductID	**txtProductID**	Product ID	(Empty string)	0, 0
TextBox	ProductName	**txtProductName**	Product Name	(Empty string)	1.1875, 0
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	(Empty string)	3, 0
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	OutputFormat = Currency	4.375, 0

5. Add the following controls to the PageFooter section:

Control	Name	Text/Caption	Misc Details	Location
Label	**lblPage**	Page	(Empty string)	0, 0
Label	**lblOf**	Of	(Empty string)	0.625, 0

TextBox	**txtPageNumber**	#	SummaryType = PageCount
			SummaryRunning = All
				0.447, 0

TextBox	**txtPageCount**	##	SummaryType = PageCount	0.875, 0

Parameters Walkthroughs

With ActiveReports, parameters may be used with simple reports to prompt a user for input before running the report or used with subreports to link a subreport to a parent report.

- o Using Parameters with Simple Reports
- o Using Parameters with Subreports

Parameters with Simple Reports

In ActiveReports, the Parameters dialog can be used to prompt for user input when reports are generated. If you add <%FieldName | PromptString | DefaultValue | Type %> to the reports SQL string, it will cause the Parameters dialog to be displayed.

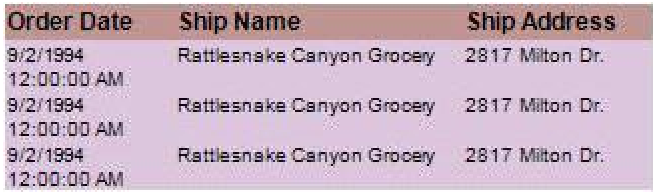
This walkthrough illustrates the basics of using parameters in simple reports.

The walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the data source to a database
- Adding controls to contain the data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Order Date	Ship Name	Ship Address
9/2/1994 12:00:00 AM	Rattlesnake Canyon Grocery	2817 Milton Dr.
9/2/1994 12:00:00 AM	Rattlesnake Canyon Grocery	2817 Milton Dr.
9/2/1994 12:00:00 AM	Rattlesnake Canyon Grocery	2817 Milton Dr.

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptParams.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "SELECT * FROM orders INNER JOIN [order details] ON orders.orderid = [order details].orderid WHERE orderdate =#<%Date|Order date:|1/1/1994|D%>#".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptParams.
2. Make the following changes to the group header:
   - o Change the name to ghOrders
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblOrderDate**	Order Date	0, 0
Label	**lblShipName**	Ship Name	1.1875, 0
Label	**lblShipAddress**	Ship Address	3.1875, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	OrderDate	**txtOrderDate**	Order Date	0, 0
TextBox	ShipName	**txtShipName**	Ship Name	1.1875, 0
TextBox	ShipAddress	**txtShipAddress**	Ship Address	3.1875, 0

Parameters with Subreports

Parameters can be used with subreports to connect the subreport to the parent report. By setting a parameter for the field that links the parent report to the subreport, the parent report can pass the information to the subreport through the parameters.

Note Subreports will not render PageHeader/Footer sections.

This walkthrough illustrates how to setup a subreport using parameters to link the parent report to the subreport.

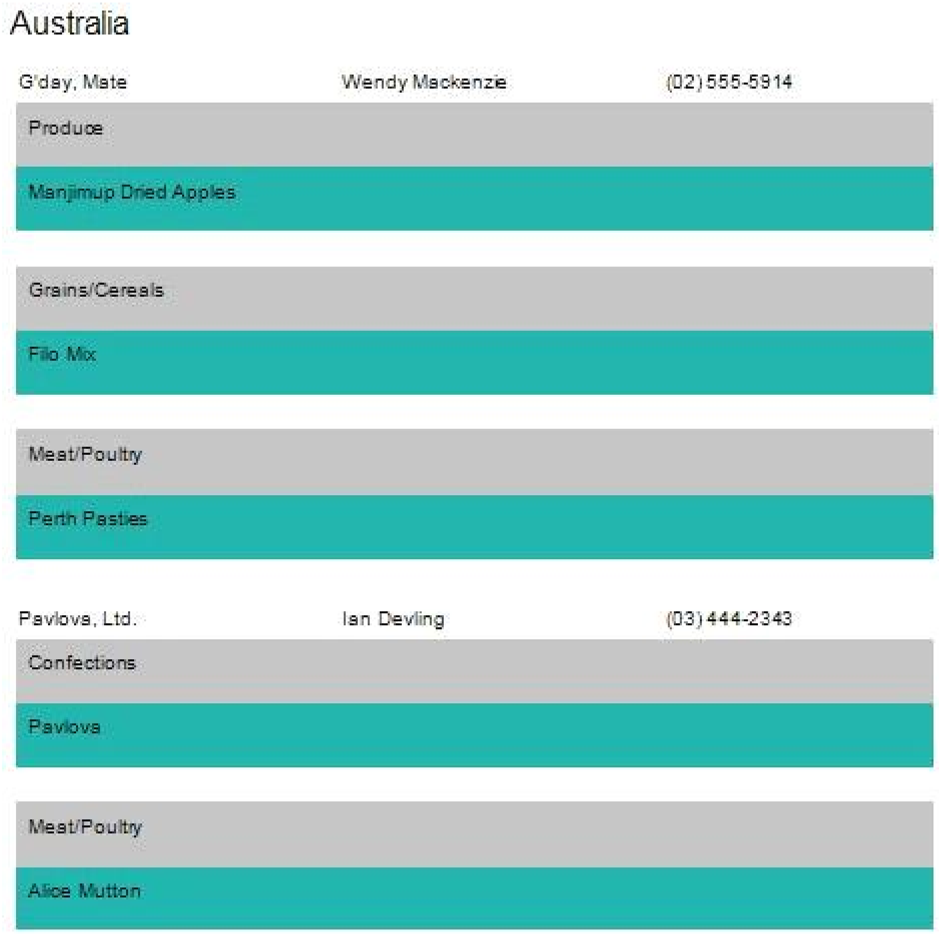
This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the parent report to a data source
- Connecting the child report to a data source using parameters
- Adding controls to display the data
- Adding the code needed to link the subreport to the current record's supplierID

- Adding the code to set the subreport's ShowParametersUI property to False

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Australia

G'day, Mate	Wendy Mackenzie	(02) 555-5914
Produce		
Manjimup Dried Apples		
Grains/Cereals		
Filo Mix		
Meat/Poultry		
Perth Pasties		
Pavlova, Ltd.	Ian Devling	(03) 444-2343
Confections		
Pavlova		
Meat/Poultry		
Alice Mutton		

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptParent.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptChild.
7. Click **Open**.

Connecting the parent report to a data source

To connect the parent report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from suppliers order by country".
7. Click **OK** to return to the report design surface.

Connecting the child report to a data source using parameters

To connect the child report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "SELECT * FROM products INNER JOIN categories ON products.categoryid = categories.categoryid WHERE products.supplierID =<%SupplierID%>".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptParent.
2. Make the following changes to the group header:
   - Change the name to ghSuppliers
   - Change the DataField property to Country
3. Add the following controls to rptParent, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
TextBox	Country	**txtCountry**	Country	GroupHeader	0, 0
TextBox	CompanyName	**txtCompanyName**	Company Name	Detail	0.0625, 0.0625
TextBox	ContactName	**txtContactName**	Contact Name	Detail	2.312, 0.0625
TextBox	Phone	**txtPhone**	Phone	Detail	4.562, 0.0625
Subreport	(Empty string)	**Subreport1**	(Empty string)	Detail	0.0625, 0.312

4. Add a GroupHeader/Footer section to rptChild.
5. Make the following changes to the group header:
   - Change the name to ghProducts
   - Change the DataField property to CategoryName
6. Add the following controls to rptChild, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
TextBox	CategoryName	**txtCategoryName**	Category Name	GroupHeader	0.0625, 0.0625
TextBox	ProductName	**txtProductName**	Product Name	Detail	0.0625, 0.0625

Adding the code needed to link the subreport to the current record's supplierID

To write the code in Visual Basic

- Right-click in any section of the design window of rptParent, and click on **View Code** to display the code view for the report. At the top left of the code view for rptParent, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptParent's Detail_Format event. Add code to the handler to:
  - Link the subreport to the current record's supplierID

To write the code in C#

- Click on the Detail section of rptParent to select the section. Click on the events icon in the **Properties** window to display available events for the report. Double-click *Format*. This creates an event-handling method for rptParent's Detail_Format event. Add code to the handler to:
  - Link the subreport to the current record's supplierID

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal _
          e As System.EventArgs) Handles Detail.Format
        Dim rpt As New rptChild()
        Me.SubReport1.Report = rpt
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
          rptChild rpt = new rptChild();
          this.SubReport1.Report = rpt;
}
```

Adding the code to set the subreport's ShowParametersUI property to False

To write the code in Visual Basic

- Right-click in any section of the design surface of rptChild, and click on **View Code** to display the code view for the report. At the top left of the code view for rptChild, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for the rptChild's ReportStart event. Add code to the handler to:

    - Set the subreport's ShowParametersUI property to False

To write the code in C#

- Click in the gray area below rptChild to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for rptChild's ReportStart event. Add code to the handler to:

    - Set the subreport's ShowParametersUI property to False

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptChild_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
        Me.ShowParameterUI = False
End Sub

[C#]

private void rptChild_ReportStart(object sender, System.EventArgs
         eArgs)
{
         this.ShowParameterUI = false;
}
```

Printing Walkthroughs

With ActiveReports, printer settings can be modified at design time as well as at run time. The following walkthroughs will describe how to make such modifications in your report.

- Duplexing
- Multiple Copies
- Orientation
- Scaling Pages

Duplexing

This walkthrough illustrates how to set the type of duplex action to use when printing out double-sided reports.

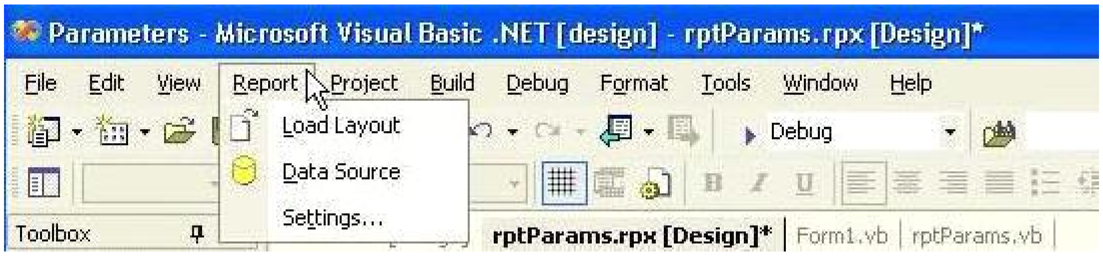
This walkthrough is split up into the following activities:

- Accessing the printer settings dialog
- Setting the type of duplexing for printing double-sided reports
- Using code to set the type of duplexing for printing

Accessing the printer settings dialog

To access the printer settings dialog

1. Open an existing ActiveReport.
2. Click on any section of the report to select it.
3. Click on **Report > Settings...**



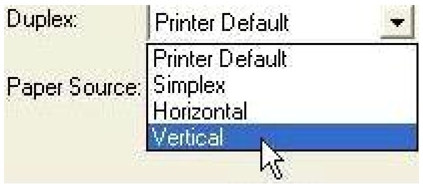
4. Click on Printer Settings.

Setting the type of duplexing for printing double-sided reports

To set the type of duplexing for printing double-sided reports

1. Go to the printer settings dialog.
2. For **Duplex**, choose one of the four options:

Printer default: the report will use the default setting on the selected printer.
Simplex: turns off duplex printing.
Horizontal: prints horizontally on both sides of the paper.
Vertical: prints vertically on both sides of the paper.



Using code to set the type of duplexing at run time

To write the code in Visual Basic

- Right-click in any section of the design window of your report, and click on **View Code** to display the code view for the report. At the top left of the code view for the report, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

  - Set the type of duplexing needed in the report

To write the code in C#

- Click in the gray section underneath the report to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

- Set the type of duplexing needed in the report

The following example shows what the code for the method looks like for setting horizontal duplexing:

```
[Visual Basic]

Private Sub rptKeepTG_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
         Me.PageSettings.Duplex = Drawing.Printing.Duplex.Horizontal
End Sub

[C#]

private void rptprint_ReportStart(object sender, System.EventArgs  eArgs)
{
         this.PageSettings.Duplex = System.Drawing.Printing.
                  Duplex.Horizontal
}
```

Multiple Copies

This walkthrough illustrates how to set multiple copies for printing a report.

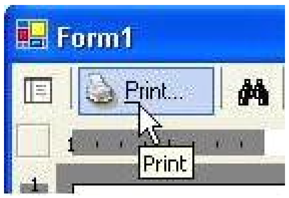
This walkthrough is split up into the following activities:

- Setting multiple copies for printing a report from the print dialog at run time
- Setting multiple copies in code for printing a report
- Using code to set multiple copies for printing

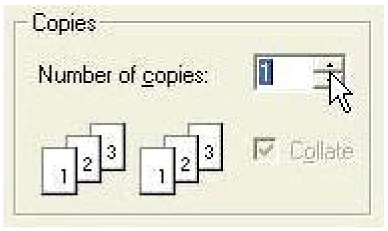
Setting multiple copies for printing a report from the print dialog at run time

To set multiple copies for printing a report from the print dialog at run time

1. Press F5 to run an existing ActiveReport.
2. Click on the printer icon in the viewer window.



3. In the Copies box, select the number of copies needed.



Using code to set multiple copies for printing at run time

To write the code in Visual Basic

- Right-click in any section of the design window of your report, and click on **View Code** to display the code view for the report. At the top left of the code view for the report, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

  - Set multiple copies of the report for printing

To write the code in C#

- Click in the gray section underneath the report to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

  - Set multiple copies of the report for printing

The following example shows what the code for the method looks like for printing five copies:

```
[Visual Basic]

Private Sub rptPrint_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
         Me.Document.Printer.PrinterSettings.Copies = 5
         Me.Document.Print(false, false)
End Sub

[C#]

private void rptPrint_ReportStart(object sender, System.EventArgs  eArgs)
{
         this.Document.Printer.PrinterSettings.Copies = 5;
         this.Document.Print(false, false);
}
```

Orientation

This walkthrough illustrates how to make simple modifications to the page orientation of your report for printing.

This walkthrough is split up into the following activities:

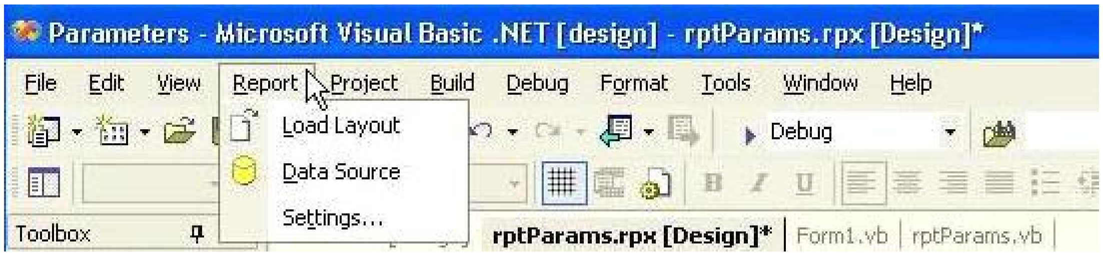
- Accessing the printer settings dialog
- Changing the page orientation of your report for printing
- Using code to change the page orientation for printing

Accessing the printer settings dialog

To access the printer settings dialog

1. Open an existing ActiveReport.

2. Click on any section of the report to select it.
3. Click on **Report > Settings...**

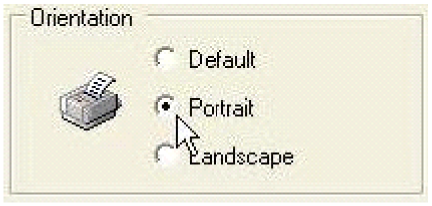

Parameters - Microsoft Visual Basic .NET [design] - rptParams.rpx [Design]*
File Edit View Report Project Build Debug Format Tools Window Help
Load Layout
Data Source
Settings...
Debug
Toolbox
rptParams.rpx [Design]*
Form1.vb
rptParams.vb

4. Click on Printer Settings.

Changing the page orientation of your report for printing

To change the page orientation of your report for printing

1. Go to the printer settings dialog.
2. In the orientation box, select either portrait or landscape.



3. Click **OK** to return to the report.

   Note Page orientation can only be modified before the report runs. Otherwise, changes made to the page orientation will not be reflected when printed.

Using code to change the page orientation

To write the code in Visual Basic

- Right-click in any section of the design window of your report, and click on **View Code** to display the code view for the report. At the top left of the code view for the report, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

  - Change the page orientation of the report for printing

To write the code in C#

- Click in the gray section underneath the report to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for the report's ReportStart event. Add code to the handler to:

  - Change the page orientation of the report for printing

The following example shows what the code for the method looks like for changing the page orientation to landscape:

```
[Visual Basic]

Private Sub rptPrint_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
         Me.PageSettings.Orientation = PageOrientation.Landscape
End Sub
[C#]

private void rptPrint_ReportStart(object sender, System.EventArgs  eArgs)
{
         this.PageSettings.Orientation = DataDynamics.ActiveReports.
                  Document.PageOrientation.Landscape;
}
```

Scaling Pages

This walkthrough illustrates how to set scaling to print a report.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Windows Form to set scaling

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptScale.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from categories INNER JOIN products ON categories.categoryID = products.categoryID order by products.categoryid".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptScale.
2. Make the following changes to the group header:
   - Change the name to ghProducts
   - Change the DataField property to CategoryName
   - Change the GroupKeepTogether property to FirstDetail
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Location
TextBox	CategoryName	**txtCategoryName**	Category Name	0, 0
Label	(Empty string)	**lblProductName**	Product Name	0, 0.3125
Label	(Empty string)	**lblUnitsInStock**	Units In Stock	2.125, 0.3125
Label	(Empty string)	**lblUnitsOnOrder**	Units On Order	3.25, 0.3125
Label	(Empty string)	**lblUnitPrice**	Unit Price	4.375, 0.3125

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	2.125, 0	(Empty string)
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	3.25, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4.375, 0	Currency

Adding code to the form to set scaling

To write the code in Visual Basic

- Right-click in any section of the design window of Form1, and click on **View Code** to display the code view for the form. Add code to the Form to set scaling.

To write the code in C#

- Double-click at the top of Form1 to see the Code View for the form. Add code to the Form to set scaling.

The following example shows what the code looks like:

```
[Visual Basic]

Dim i As Integer
Dim rpt As New rptScale()
Dim m_myARPrinter As New DataDynamics.ActiveReports._
         Interop.SystemPrinter()
Private Sub Form1_Load(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles MyBase.Load
    Viewer1.Document = rpt.Document
    rpt.Run()
    arScale()
End Sub
```

```
Private Sub arScale()
         m_myARPrinter.StartJob("Test Printer")

         Dim aPage As New DataDynamics.ActiveReports.Document.Page()
         Dim rec As New System.Drawing.RectangleF()
         Dim xOffSet As Single
         Dim yOffSet As Single
         Dim adjustedWidth As Single
         Dim xPos As Single

         xOffSet = m_myARPrinter.PhysicalOffsetX / _
                  m_myARPrinter.Graphics.DpiX
         yOffSet = m_myARPrinter.PhysicalOffsetY / _
                  m_myARPrinter.Graphics.DpiY
         adjustedWidth = (aPage.Width / 3) - (xOffSet / 2)
         xPos = 0
         Dim nCount As Integer
         nCount = rpt.Document.Pages.Count
         m_myARPrinter.StartPage()

         For i = 0 To nCount - 1
                  aPage = rpt.Document.Pages(i)
                  m_myARPrinter.Graphics.PageUnit = GraphicsUnit.Pixel
                  rec = System.Drawing.RectangleF.FromLTRB(xOffSet + _
                           xPos, yOffSet, (xOffSet + xPos) + _
                           adjustedWidth, yOffSet + adjustedWidth)
                  xPos = adjustedWidth + xPos
                  aPage.Draw(m_myARPrinter.Graphics, rec)
         Next
         m_myARPrinter.EndPage()
         m_myARPrinter.EndJob()
End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{

         rpt = new ActiveReport1();
         this.viewer1.Document = rpt.Document;
         rpt.Run();
         arScale();
}

ActiveReport1 rpt;
DataDynamics.ActiveReports.Document.Page aPage;
private void arScale()
{
         aPage = new DataDynamics.ActiveReports.Document.Page();
         DataDynamics.ActiveReports.Interop.SystemPrinter m_myARPrinter =
                  new DataDynamics.ActiveReports.Interop.SystemPrinter();

         m_myARPrinter.StartJob("Test Printer");

         System.Drawing.RectangleF rec;
         float xOffSet = m_myARPrinter.PhysicalOffsetX/
                  m_myARPrinter.Graphics.DpiX;
         float yOffSet = m_myARPrinter.PhysicalOffsetY/
                  m_myARPrinter.Graphics.DpiY;
         float adjustedWidth = (aPage.Width/3)-(xOffSet*2);
         float xPos = 0;
         int nCount = rpt.Document.Pages.Count;
         m_myARPrinter.StartPage();
         for(int i=0; i < nCount; i++)
         {
                  aPage = rpt.Document.Pages[i];
                  m_myARPrinter.Graphics.PageUnit = System.
                           Drawing.GraphicsUnit.Pixel;
                  rec = System.Drawing.RectangleF.FromLTRB
                           (xOffSet+xPos, yOffSet,(xOffSet+xPos)+
                              adjustedWidth,yOffSet+adjustedWidth);
                    xPos = adjustedWidth + xPos;
                    aPage.Draw(m_myARPrinter.Graphics,rec);
          }
          m_myARPrinter.EndPage();
          m_myARPrinter.EndJob();
}
```

Rich Text and Field Merging

ActiveReports supports field merged reports using the RichText control. The RichText control can contain field placeholders that are replaceable with their values (merged) at run time.

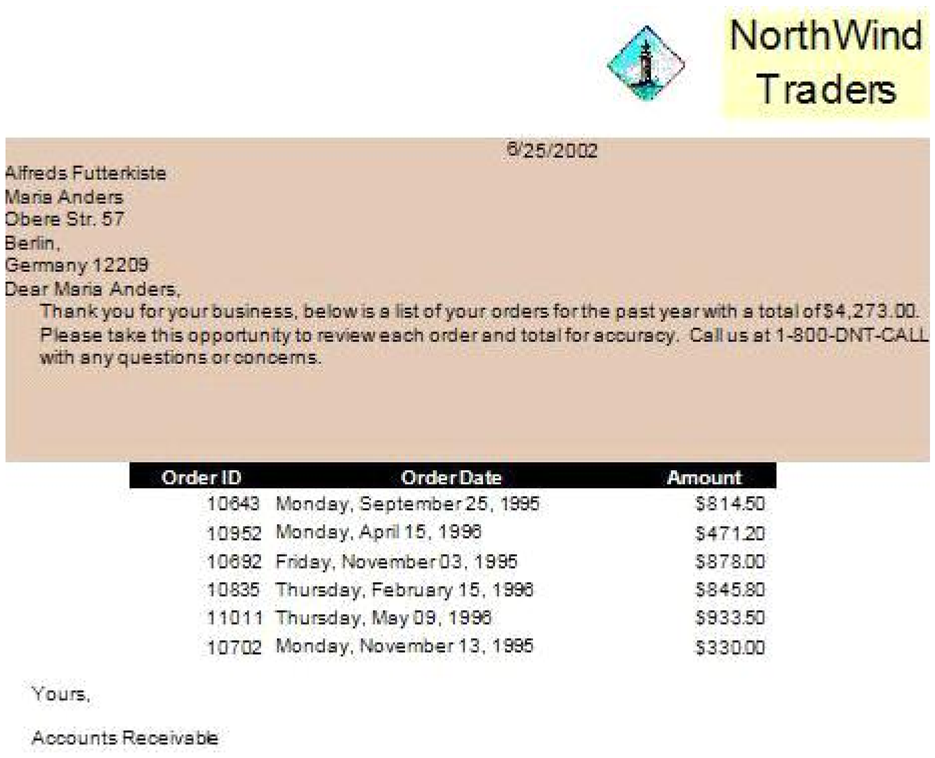
This walkthrough illustrates how to create a mail-merge report using the RichText control.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to the Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding fields to the RichText control
- Using the FetchData event to get information from the data source
- Adding code to update field values in the RichText control for each record
- Adding code to the group header BeforePrint event

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you complete this walkthrough, you will have a report that looks similar to the following.



NorthWind Traders

6/25/2002

Alfreds Futterkiste
Maria Anders
Obere Str. 57
Berlin,
Germany 12209
Dear Maria Anders,
Thank you for your business, below is a list of your orders for the past year with a total of $4,273.00. Please take this opportunity to review each order and total for accuracy. Call us at 1-800-DNT-CALL with any questions or concerns.

Order ID	OrderDate	Amount
10643	Monday, September 25, 1995	$814.50
10952	Monday, April 15, 1996	$471.20
10692	Friday, November 03, 1995	$878.00
10835	Thursday, February 15, 1996	$845.80
11011	Thursday, May 09, 1996	$933.50
10702	Monday, November 13, 1995	$330.00

Yours,

Accounts Receivable

Adding an ActiveReport to the Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptLetter.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "SELECT
   Customers.CustomerID, Customers.CompanyName,
   Customers.ContactName, Customers.Address,
   Customers.City, Customers.Region, Customers.Country, Customers.PostalCode,
   Orders.OrderID, Orders.OrderDate, [Order Subtotals].Subtotal
   FROM Customers INNER JOIN ([Order Subtotals] INNER JOIN Orders ON [Order Subtotals].OrderID = Orders.OrderID) ON Customers.CustomerID = Orders.CustomerID".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptLetter.
2. Make the following changes to the group header:
   - Change the name to ghCustomerID
   - Change the DataField property to CustomerID
   - Change the ColumnLayout property to False
   - Change KeepTogether property to True
3. Make the following changes to the group footer:
   - Change the name to gfCustomerID
   - Change the ColumnLayout property to False
   - Change KeepTogether property to True
   - Change the KeepTogether property for the group header and group footer to True.
4. Add the following controls to the PageHeader section:

Control	Name	Text/Caption	Location
Picture	**imgLogo**	(Empty string)	4, 0
Label	**lblNorthWind**	NorthWind	5, 0.0625
Label	**lblTraders**	Traders	5, 0.4375

5. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Misc Details	Location
RichText	(Empty	**rtf**	(Empty	(Empty string)	0, 0

	string)		string)		
Textbox	Subtotal	**txtTotalOrders**	(Empty string)	Visible = False OutputFormat = Currency SummaryType = SubTotal SummaryGroup = ghCustomerID	5.437, 0.0625
Label	(Empty string)	**lblOrderID**	Order ID	(Empty string)	0.875, 2.25
Label	(Empty string)	**lblOrderDate**	Order Date	(Empty string)	1.875, 2.25
Label	(Empty string)	**lblAmount**	Amount	(Empty string)	4.375, 2.25

6. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Textbox	OrderID	**txtOrderID**	Order ID	Align: right	0.8125, 0
Textbox	OrderDate	**txtOrderDate**	Order Date	OutputFormat = Long Date	1.875, 0
Textbox	Subtotal	**txtSubtotal**	Subtotal	OutputFormat = Currency Align: right	4.3125, 0

7. Add the following controls to the GroupFooter section:

Control	Name	Text/Caption	Location
Label	**lblYours**	Yours,	0.1875, 0.125
Label	**lblNTAP**	Accounts Receivable	0.1875, 0.4375

8. Add the following controls to the PageFooter:

Control	Name	Text/Caption	Location
Label	**lblNWAddress**	NorthWind Traders, One Portals Way, Twin Points WA 98156	0, 0.0625

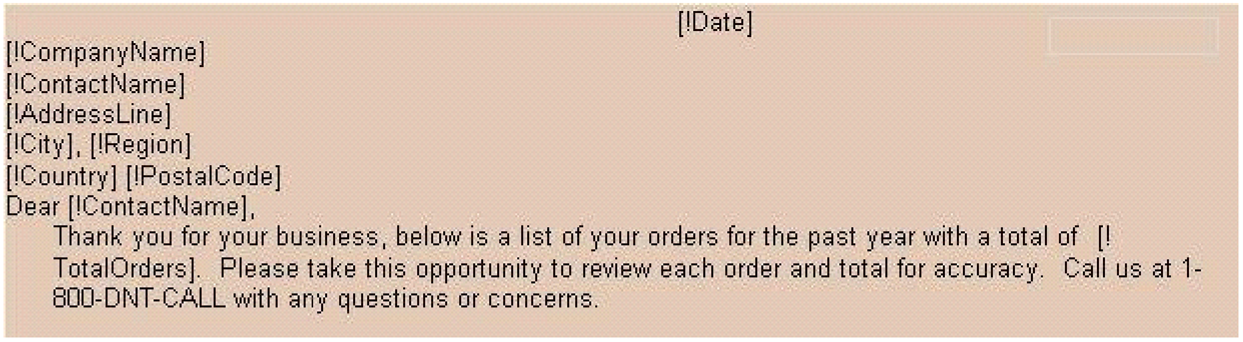
Adding fields to the RichText control

To add fields to the RichText control

1. Double-click inside the RichText control box to select it.
2. Right-click inside the box and choose **Insert Field**.
3. Insert the following fields in the following order:
   - o Date
   - o CompanyName
   - o ContactName
   - o AddressLine
   - o City
   - o Region
   - o Country
   - o PostalCode
   - o ContactName
   - o TotalOrders
4. Add the following text to the RichText control box: "Dear [!ContactName], Thank you for your business, below is a list of your orders for the past year with a total of [!TotalOrders].

Please take this opportunity to review each order and total for accuracy. Call us at 1-800-DNT-CALL with any questions or concerns.

5. Your RichText control should be arranged like the following.



[!Date]
[!CompanyName]
[!ContactName]
[!AddressLine]
[!City], [!Region]
[!Country] [!PostalCode]
Dear [!ContactName],
Thank you for your business, below is a list of your orders for the past year with a total of [!TotalOrders]. Please take this opportunity to review each order and total for accuracy. Call us at 1-800-DNT-CALL with any questions or concerns.

Using the FetchData event to get information from the data source

To write the code in Visual Basic

- Right-click in any section of the design window of rptLetter, and click on **View Code** to display the code view for the report. At the top left of the code view for rptLetter, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptLetter's FetchData event. Add code to the handler to:
  - Retrieve information from the data source

To write the code in C#

- Click in the gray area below rptLetter to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptLetter's FetchData event. Add code to the handler to:
  - Retrieve information from the data source

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_companyName As String
Dim m_contactName As String
Dim m_addressLine As String
Dim m_city As String
Dim m_region As String
Dim m_country As String
Dim m_postalCode As String
Private Sub rptLetter_FetchData(ByVal sender As Object, ByVal _
         eArgs As DataDynamics.ActiveReports. _
         ActiveReport.FetchEventArgs) Handles MyBase.FetchData
        m_companyName = Fields("CompanyName").Value
        m_contactName = Fields("ContactName").Value
        m_addressLine = Fields("Address").Value
        m_city = Fields("City").Value
        If Fields("Region").Value Is System.DBNull.Value Then
            m_region = ""
        Else
            m_region = Fields("Region").Value
        End If
        m_country = Fields("Country").Value
        m_postalCode = Fields("PostalCode").Value
End Sub

[C#]

string m_companyName;
string m_contactName;
string m_addressLine;
string m_city;
string m_region;
string m_country;
string m_postalCode;
private void rptLetter_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
         m_companyName = Fields["CompanyName"].Value.ToString();
         m_contactName = Fields["ContactName"].Value.ToString();
         m_addressLine = Fields["Address"].Value.ToString();
         m_city = Fields["City"].Value.ToString();
         if(Fields["Region"].Value is System.DBNull)
                  m_region = "";
         else
                  m_region = Fields["Region"].Value.ToString();
         m_country = Fields["Country"].Value.ToString();
         m_postalCode = Fields["PostalCode"].Value.ToString();
}
```

Adding code to update the field values in the Rich Text control

To write the code in Visual Basic

- Right-click in any section of the design window of rptLetter, and click on **View Code** to display the code view for the report. At the top left of the code view for rptLetter, click the drop-down arrow and select *ghCustomerID*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptLetter's ghCustomerID_Format event. Add code to the handler to:
  - Update the field values in the RichText control

To write the code in C#

- Click in the GroupHeader section of rptLetter to select it. Click on the events icon in the **Properties** window for ghCustomerID to display available events for the section. Double-click *Format*. This creates an event-handling method for rptLetter's ghCustomerID_Format event. Add code to the handler to:
  - Update the field values in the RichText control

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub ghCustomerID_Format(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles ghCustomerID.Format
        Me.rtf.ReplaceField("CompanyName", m_companyName)
        Me.rtf.ReplaceField("ContactName", m_contactName)
        Me.rtf.ReplaceField("AddressLine", m_addressLine)
        Me.rtf.ReplaceField("City", m_city)
        Me.rtf.ReplaceField("Region", m_region)
        Me.rtf.ReplaceField("Country", m_country)
        Me.rtf.ReplaceField("PostalCode", m_postalCode)
        Me.rtf.ReplaceField("Date", System.DateTime.Today.Date)
End Sub

[C#]

private void ghCustomerID_Format(object sender, System.EventArgs eArgs)
{
         this.rtf.ReplaceField("CompanyName", m_companyName);
         this.rtf.ReplaceField("ContactName", m_contactName);
         this.rtf.ReplaceField("AddressLine", m_addressLine);
         this.rtf.ReplaceField("City", m_city);
         this.rtf.ReplaceField("Region", m_region);
         this.rtf.ReplaceField("Country", m_country);
         this.rtf.ReplaceField("PostalCode", m_postalCode);
         this.rtf.ReplaceField("Date",
                  System.DateTime.Today.Date.ToString());
}
```

Adding code to the Group Header BeforePrint Event

To write the code in Visual Basic

- Right-click in any section of the design window of rptLetter, and click on **View Code** to display the code view for the report. At the top left of the code view for rptLetter, click the drop-down arrow and select *ghCustomerID*. At the top right of the code window, click the drop-down arrow and select *BeforePrint*. This creates an event-handling method for rptLetter's ghCustomerID_BeforePrint event.

To write the code in C#

- Click in the GroupHeader section of rptLetter to select it. Click on the events icon in the **Properties** window for ghCustomerID to display available events for the section. Double-click *BeforePrint*. This creates an event-handling method for rptLetter's ghCustomerID_BeforePrint event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub ghCustomerID_BeforePrint(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles _
         ghCustomerID.BeforePrint
        Me.rtf.ReplaceField("TotalOrders", Me.txtTotalOrders.Text)
End Sub

[C#]

private void ghCustomerID_BeforePrint(object sender, System.EventArgs
         eArgs)
{
         this.rtf.ReplaceField("TotalOrders",this.txtTotalOrders.Text);
}
```

Run-Time Reporting Walkthroughs

ActiveReports allows objects, controls and the data source to be completely accessible at run time. These properties can be modified to provide a dynamic view of your report.

- Creating and Modifying Report Layouts at Run Time
- Modifying the Report Data Source at Run Time

Creating and Modifying Report Layouts at Run Time

ActiveReports objects and controls are completely accessible at run time. The properties of any of the report sections or controls can be modified to produce a dynamic view of the report. The format event allows the properties of report sections and controls to be modified including height, visibility and other visual properties. The format event is the only event in which the printable area of a section can be modified. Once this event is completed, any changes to the section's height will not be reflected in the report output.

> **Note** Controls may be added dynamically in the ReportStart event but should not be added dynamically after the ReportStart event fires as problems may result.

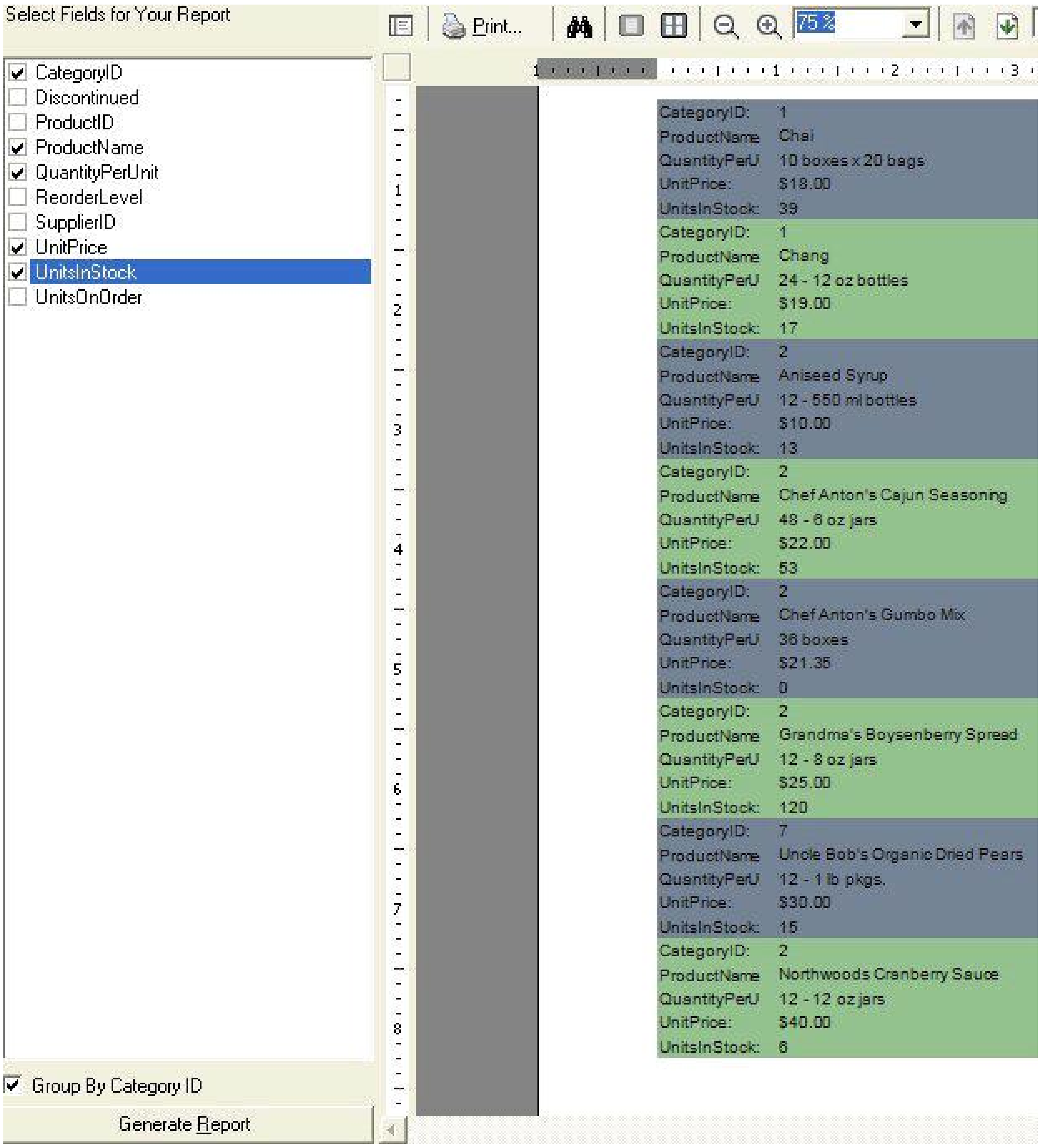
This walkthrough illustrates how to create a report layout at run time based on user input.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the Windows Form to contain data
- Adding the data adapter to the Windows Form and generating a data set
- Adding code to the blank report
- Adding code to the Windows Form
- Adding code to the report's Detail_Format event
- Adding code to the report's ReportStart event
- Adding code to the Window Form's "Generate Report" button Click event
- Adding code to the Checked ListBox's SelectedIndexChanged event on the Windows Form
- Adding code to the Form_Load event

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have output that looks similar to the following.


Select Fields for Your Report

- [✓] CategoryID
- [ ] Discontinued
- [ ] ProductID
- [✓] ProductName
- [✓] QuantityPerUnit
- [ ] ReorderLevel
- [ ] SupplierID
- [✓] UnitPrice
- [✓] UnitsInStock
- [ ] UnitsOnOrder

[✓] Group By Category ID

CategoryID:	1
ProductName	Chai
QuantityPerU	10 boxes x 20 bags
UnitPrice:	$18.00
UnitsInStock:	39
CategoryID:	1
ProductName	Chang
QuantityPerU	24 - 12 oz bottles
UnitPrice:	$19.00
UnitsInStock:	17
CategoryID:	2
ProductName	Aniseed Syrup
QuantityPerU	12 - 550 ml bottles
UnitPrice:	$10.00
UnitsInStock:	13
CategoryID:	2
ProductName	Chef Anton's Cajun Seasoning
QuantityPerU	48 - 6 oz jars
UnitPrice:	$22.00
UnitsInStock:	53
CategoryID:	2
ProductName	Chef Anton's Gumbo Mix
QuantityPerU	36 boxes
UnitPrice:	$21.35
UnitsInStock:	0
CategoryID:	2
ProductName	Grandma's Boysenberry Spread
QuantityPerU	12 - 8 oz jars
UnitPrice:	$25.00
UnitsInStock:	120
CategoryID:	7
ProductName	Uncle Bob's Organic Dried Pears
QuantityPerU	12 - 1 lb pkgs.
UnitPrice:	$30.00
UnitsInStock:	15
CategoryID:	2
ProductName	Northwoods Cranberry Sauce
QuantityPerU	12 - 12 oz jars
UnitPrice:	$40.00
UnitsInStock:	6

Adding an ActiveReport to your project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptFieldsRT.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail field. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products".
7. Click **OK** to return to the report design surface.

Adding controls to the Windows Form to contain data

To add controls to the form

- Add the following controls to the Windows Form:

Control	Dock Property	Name	Text/Caption	Misc Details
Panel	Left	**Panel1**	(Empty string)	Height: 406 Width: 200
Label	Top	**lblSelectFields**	Select Fields for Your Report	Location: Panel1
Checked ListBox	Fill	**clbFields**	Report Fields	Location: Panel1
Button	Bottom	**btnGenRep**	Generate Report	Location: Panel1
CheckBox	Bottom	**chkGroup**	Group By Category ID	Location: Panel1
Viewer	Fill	**Viewer1**	(Empty string)	Location: Windows Form

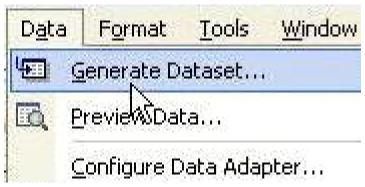
Adding a data adapter to the Windows Form and generating the dataset

To add a data adapter to the form

1. On the *Data* toolbox, double-click **OleDbDataAdapter**.
2. This adds the adapter to the Windows Form and opens the Data Adapter Configuration Wizard.
3. Follow the instructions to choose the data connection, select the data provider, select the database and enter the SQL statement, "SELECT CategoryID, Discontinued, ProductID, ProductName, QuantityPerUnit, ReorderLevel, SupplierID, UnitPrice, UnitsInStock, UnitsOnOrder FROM Products" (For help with this, see "Connecting the report to a data source" above.
4. Click "Finish" to return to the Windows Form.

To generate the dataset

1. Click on **Data** > **Generate Dataset...** on the Visual Studio menu bar.



2. This opens the Generate Dataset dialog.
3. Click **New** and make sure the selected table to be added to the data set is Products.
4. Check "Add this dataset to the designer" and click **OK**.

Adding code to the blank ActiveReport

To add code to the blank report

- Insert the following code beneath "ActiveReports Designer generated code."

```
[Visual Basic]

Private m_arrayFields As ArrayList
Public WriteOnly Property FieldsList() As ArrayList
         Set(ByVal Value As ArrayList)
                  m_arrayFields = Value
         End Set
End Property

Private m_useGroups As Boolean
Public WriteOnly Property UseGroups() As Boolean
         Set(ByVal Value As Boolean)
                  m_useGroups = False
                  m_useGroups = Value
         End Set
End Property

Private Sub constructReport()
         Try
                  Me.Detail.CanGrow = True
                  Me.Detail.CanShrink = True
                  Me.Detail.KeepTogether = True
                  If m_useGroups = True Then
                           Me.Sections.InsertGroupHF()

                           CType(Me.Sections("GroupHeader1"), GroupHeader)._
                                    DataField = "CategoryID"
                           Me.Sections("GroupHeader1").BackColor _
                                    = System.Drawing.Color.SlateBlue
                           Me.Sections("GroupHeader1").CanGrow = True
                           Me.Sections("GroupHeader1").CanShrink = True
                           CType(Me.Sections("GroupHeader1"), _
                                    GroupHeader).RepeatStyle = _
                                    RepeatStyle.OnPageIncludeNoDetail
                           Me.Sections("GroupHeader1").Height = 0

                           Dim txt As New TextBox()
                           txt.DataField = "CatagoryID"
                           txt.Location = New System.Drawing.PointF(0.0F, 0)
                           txt.Width = 2.0F
                           txt.Height = 0.3F
                           txt.Style = "font-weight: bold; font-size: 16pt"
                           Me.Sections("GroupHeader1").Controls.Add(txt)
                  End If
                  For i = 0 To m_arrayFields.Count - 1
                           If (m_useGroups = False) Or (m_useGroups _
                                    AndAlso m_arrayFields(i)._
```

```
                                        ToString <> "CategoryID") Then
                                        Dim lbl As New Label()
                                        lbl.Text = m_arrayFields(i) + ":"
                                        lbl.Location() = New _
                                                System.Drawing.PointF_
                                                (0.0F, m_currentY)
                                        lbl.Width = 0.9F
                                        lbl.Height = m_defaultHeight
                                        Me.Detail.Controls.Add(lbl)

                                        Dim txt As New TextBox()
                                        txt.DataField = m_arrayFields(i)
                                        txt.Location = New _
                                                System.Drawing.PointF_
                                                (1.0F, m_currentY)
                                        txt.Width = m_defaultWidth
                                        txt.Height = m_defaultHeight
                                        Me.Detail.Controls.Add(txt)
                                        If m_arrayFields(i) = "UnitPrice" Then
                                                txt.OutputFormat = "$#.00"
                                        End If
                                        m_currentY = m_currentY + m_defaultHeight
                                End If
                        Next
        Catch ex As Exception
                System.Windows.Forms.MessageBox.Show("Error _
                        in Report-constructReport: _
                        " + ex.Message, "Project Error", _
                        System.Windows.Forms.MessageBoxButtons.OK, _
                        System.Windows.Forms.MessageBoxIcon.Error)
        End Try
End Sub

[C#]

private ArrayList m_arrayFields;
public ArrayList FieldsList
{
        set{m_arrayFields = value;}
}

private bool m_useGroups = false;
public bool UseGroups
{
        set{m_useGroups = value;}
}

float m_defaultHeight = .2f;
float m_defaultWidth = 4f;
float m_currentY = 0f;

private void constructReport()
{
        try
        {
                this.Detail.CanGrow = true;
                this.Detail.CanShrink = true;
                this.Detail.KeepTogether = true;

                if(m_useGroups)
                {
                        this.Sections.InsertGroupHF();
                        ((GroupHeader)this.Sections["GroupHeader1"]).
                                DataField = "CategoryID";
                        this.Sections["GroupHeader1"].BackColor =
                                System.Drawing.Color.SlateBlue;
                        this.Sections["GroupHeader1"].CanGrow = true;
                        this.Sections["GroupHeader1"].CanShrink = true;
                        ((GroupHeader)this.Sections["GroupHeader1"]).
                                RepeatStyle = RepeatStyle.
                                OnPageIncludeNoDetail;
```

```
                              this.Sections["GroupFooter1"].Height = 0;

                              TextBox txt = new TextBox();
                              txt.DataField = "CategoryID";
                              txt.Location = new System.Drawing.PointF(0f,0);
                              txt.Width =2f;
                              txt.Height = .3f;
                              txt.Style = "font-weight: bold;
                                       font-size: 16pt;";
                              this.Sections["GroupHeader1"].Controls.Add(txt);
                    }
                    for(int i=0;i<m_arrayFields.Count;i++)
                    {
                              if(!m_useGroups || (m_useGroups &&
                                       m_arrayFields[i].ToString()
                                       != "CategoryID"))
                              {
                                       Label lbl = new Label();
                                       lbl.Text = m_arrayFields[i].ToString() +
                                                ":";
                                       lbl.Location = new
                                                System.Drawing.PointF
                                                (0f,m_currentY);
                                       lbl.Width =.9f;
                                       lbl.Height = m_defaultHeight;
                                       this.Detail.Controls.Add(lbl);

                                       TextBox txt = new TextBox();
                                       txt.DataField = m_arrayFields
                                                [i].ToString();
                                       txt.Location = new System.Drawing.
                                                PointF(1f,m_currentY);
                                       txt.Width =m_defaultWidth;
                                       txt.Height = m_defaultHeight;
                                       this.Detail.Controls.Add(txt);
                                       m_currentY = m_currentY +
                                                m_defaultHeight;
                              }
                    }
          }
          catch(Exception ex)
          {
                    System.Windows.Forms.MessageBox.Show("Error in Report-
                              constructReport: " + ex.Message,"Project
                              Error",System.Windows.Forms.
                              MessageBoxButtons.OK,System.
                              Windows.Forms.MessageBoxIcon.Error);
          }
}
```

Adding code to the Windows Form

To add code to the form

- Insert the following code beneath "ActiveReports Designer generated code."

```
[Visual Basic]

 Private Sub fillCheckBox()
        For i = 0 To Me.DataSet11.Tables.Count - 1
            For c = 0 To Me.DataSet11.Tables(i).Columns.Count - 1
                Me.clbFields.Items.Add(Me._
                    DataSet11.Tables(i).Columns(c)._
                ColumnName)
            Next
        Next
 End Sub
 Private Sub launchReport()
        Dim rpt As New rptFieldsRT()
        Try
            rpt.FieldsList = m_arrayField
            rpt.ds.SQL = Me.OleDbDataAdapter1.SelectCommand.CommandText
            rpt.ds.ConnectionString = Me.OleDbConnection1._
                   ConnectionString
            Viewer1.Document = rpt.Document
            rpt.Run()
        Catch ex As Exception
            System.Windows.Forms.MessageBox.Show(Me, _
                   "Error in launchReport: " _
                   + ex.Message, "Project Error", MessageBoxButtons._
                   OK, MessageBoxIcon.Error)
        End Try
 End Sub

[C#]

static void Main()
{
         Application.Run(new Form1());
}
private string getDatabasePath()
{
                   RegistryKey regKey = Registry.LocalMachine;
                   regKey = regKey.CreateSubKey("SOFTWARE\\Data
                            Dynamics\\ActiveReports.NET
                            \\SampleDB");
                   return (string)regKey.GetValue("");
}

private void fillCheckBox()
{
         for(int i=0;i<this.dataSet11.Tables.Count;i++)
         {
                   for(int c=0;c<this.dataSet11.Tables
                            [i].Columns.Count;c++)
                   {
                            this.clbFields.Items.Add(this.dataSet11.Tables[i].Columns[c].
                   ColumnName);
                   }
         }
}
ArrayList m_arrayField = new ArrayList();

private void launchReport()
{
         try
         {
                   rptFieldsRT rpt = new rptFieldsRT();
                   rpt.FieldsList = m_arrayField;
                   rpt.ds.SQL = this.oleDbDataAdapter1.
                            SelectCommand.CommandText;
                   rpt.ds.ConnectionString = this.oleDbConnection1.
                            ConnectionString;
                   this.viewer1.Document = rpt.Document;
                   rpt.Run();
         }
         catch(Exception ex)
         {
                   MessageBox.Show(this,"Error in
                            launchReport: " + ex.Message,"Project
                            Error",MessageBoxButtons.OK,MessageBoxIcon.Error);
         }
}
```

Adding code to the Detail_Format event

To write the code in Visual Basic

- Right-click in any section of the design window of rptFieldsRT, and click on **View Code** to display the code view for the report. At the top left of the code view for rptFieldsRT, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptFieldsRT's Detail_Format event.

To write the code in C#

- Click in the Detail section of rptFieldsRT to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptFieldsRT's Detail_Format event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_defaultHeight As Single = 0.2
Dim m_defaultWidth As Single = 4
Dim m_currentY As Single = 0
Dim i As Integer
Dim m_count As Integer
Private Sub Detail_Format(ByVal sender As Object, _
         ByVal e As System.EventArgs) _
         Handles Detail.Format
         If m_count Mod 2 = 0 Then
                  Me.Detail.BackColor = System.Drawing.Color.SlateGray
         Else
                  Me.Detail.BackColor = System.Drawing.Color.DarkSeaGreen
         End If
         m_count = m_count + 1
End Sub

[C#]

int m_count;
private void Detail_Format(object sender, System.EventArgs eArgs)
{
         if(m_count % 2 ==0)
         {
                  this.Detail.BackColor = System.Drawing.Color.SlateGray;
         }
         else
         {
                  this.Detail.BackColor = System.Drawing.
                           Color.DarkSeaGreen;
         }
         m_count++;
}
```

Adding code to the ReportStart event

To write the code in Visual Basic

- Right-click in any section of the design window of rptFieldsRT, and click on **View Code** to display the code view for the report. At the top left of the code view for rptFieldsRT, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for rptFieldsRT's ReportStart event.

To write the code in C#

- Click in the gray area below rptFieldsRT to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for rptFieldsRT's ReportStart event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptFieldsRT_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
         constructReport()
End Sub

[C#]

private void rptFieldsRT_ReportStart(object sender,
         System.EventArgs eArgs)
{
         constructReport();
}
```

Adding code for btnGenRep's Click event

To write the code in Visual Basic

- Double-click on btnGenRep. This creates an event-handling method for btnGenRep's Click event. Add the following code to the btnGenRep_Click event.

To write the code in C#

- Double-click on btnGenRep. This creates an event-handling method for btnGenRep's Click event. Add the following code to the btnGenRep_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub btnGenRep_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs)Handles btnGenRep.Click
        Me.m_arrayField.Clear()
        For i = 0 To Me.clbFields.CheckedItems.Count - 1
            m_arrayField.Add(Me.clbFields.CheckedItems(i).ToString)
        Next
        launchReport()
 End Sub

[C#]

private void btnGenRep_Click(object sender, System.EventArgs e)
{
         this.m_arrayField.Clear();
         for(int i=0;i<this.clbFields.CheckedItems.Count;i++)
         {
                  m_arrayField.Add(this.clbFields.CheckedItems[i].ToString());
         }
         launchReport();
}
```

Adding code to the clbFields_SelectedIndexChanged event

To write the code in Visual Basic

- Right-click in any section of the Windows Form, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for the form, click the drop-down arrow and select *clbFields*. At the top right of the code window, click the drop-down arrow and select *SelectedIndexChanged*. This creates an event-handling method for the Form1_SelectedIndexChanged event.

To write the code in C#

- Click in the designer window of the Windows Form to select it. Click on the events icon in the **Properties** window to display available events for the section. Double-click *SelectedIndexChanged*. This creates an event-handling method for the Form1_SelectedIndexChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Private Sub clbFields_SelectedIndexChanged(ByVal sender _
         As Object, ByVal e As System.EventArgs) _
         Handles CheckedListBox1.SelectedIndexChanged
        If Me.clbFields.CheckedItems.Count < 0 Then
            Me.btnGenRep.Enabled = False
        Else
            Me.btnGenRep.Enabled = True
        End If
 End Sub

[C#]

private void clbFields_SelectedIndexChanged(object
         sender, System.EventArgs e)
{
         if(this.clbFields.CheckedItems.Count>0)
         {
                  this.btnGenRep.Enabled = true;
         }
         else
         {
                  this.btnGenRep.Enabled = false;
         }
}
```

Adding code to the Form1_Load event

To write the code for the viewer in Visual Basic

- Right-click on Form1, and click on **View Code** to display the code view for the form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

    - Format the viewer to show the report when it is run

To write the code for the viewer in C#

- Click on the blue section at the top of Form1 to select the form. Click on the events icon in the **Properties** window to display available events for Form1. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:
    - Format the viewer to show the report when it is run

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Dim i As Integer
 Dim c As Integer
 Dim m_arrayField As New ArrayList()
 Private Sub Form1_Load(ByVal sender As System.Object, _
          ByVal e As System.EventArgs)Handles MyBase.Load
        Try
            Me.OleDbConnection1.ConnectionString _
                   = "Provider=Microsoft.Jet.OLEDB.4.0;_
                   Data Source=C:\Program Files\Data _
                   Dynamics\ActiveReports.NET\Data\NWIND._
                   MDB;Persist Security Info=False"
            Me.OleDbDataAdapter1.Fill(Me.DataSet11)
            fillCheckBox()
        Catch ex As Exception
            System.Windows.Forms.MessageBox.Show(Me, _
                   "Error in Form_Load: " + ex.Message, _
                   "Project Error", MessageBoxButtons.OK,_
                   MessageBoxIcon.Error)
        End Try
 End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         try
         {
                  string m_dbPath = getDatabasePath();
                  this.oleDbConnection1.ConnectionString =
                           "Provider=Microsoft.Jet.OLEDB.4.0;
                           Data Source=" + m_dbPath + "\\NWIND.MDB;Persist
                           Security Info=False";
                  this.oleDbDataAdapter1.Fill(this.dataSet11);
                  fillCheckBox();
         }
         catch(Exception ex)
         {
                  MessageBox.Show(this,"Error in Form_Load: " +
                           ex.Message,"Project Error",
                           MessageBoxButtons.OK,MessageBoxIcon.Error);
         }
}
```

Modifying the Report Data Source at Run Time

ActiveReports allows you to change the data source of a report at run time based on the location of the sample database file on the user's computer.

This walkthrough illustrates how to set up a report to set a report's data source at run time.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source

- Adding controls to the report to contain data
- Adding code to set the database path
- Adding code to the ReportStart event to change data source at run time

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

Adding an ActiveReport to your project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptModifyDS.
4. Click **Open**.

Connecting the data source to a database

To connect the data source to a database

1. Click on the yellow report DataSource icon in the Detail field. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select "Microsoft Jet 4.0 OLE DB Provider" and click **Next >>**.
4. Click on the ellipsis to browse for the access path to Nwind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products".
7. Click **OK** to return to the report design surface.

Adding controls to contain data

To add controls to the report

- Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductID	**txtProductID**	Product ID	0, 0.625	(Empty string)
TextBox	ProductName	**txtProductName**	Product Name	1.125, 0.0625	(Empty string)
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	2.25, 0.0625	(Empty string)
TextBox	UnitsOnOrder	**txtUnitsOnOrder**	Units On Order	3.375, 0.0625	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4.5, 0.0625	Currency

Adding code to set the database path

To write the code in Visual Basic

- Right-click in any section of the design window of rptModifyDS, and click on **View Code** to display the code view for the report. Beneath "ActiveReports Designer Generated

Code", type the following: "Private Function getDatabasePath() As String" and hit "Enter." This creates a function for getDatabasePath.

To write the code in C#

- Click in the gray area below rptModifyDS and click on **View Code** to display the code view for the report. Beneath "ActiveReports Designer Generated Code", type the following: "private string getDatabasePath()". This creates a function for getDatabasePath.

The following example shows what the code for the function looks like:

```
[Visual Basic]

Private Function getDatabasePath() As String
Dim regKey As RegistryKey
         regKey = Registry.LocalMachine
         regKey = regKey.CreateSubKey("SOFTWARE\\_
                  Data Dynamics\\ActiveReports.NET\\SampleDB")
         getDatabasePath = CType(regKey.GetValue(""), String)
End Function

[C#]

private string getDatabasePath()
{
         RegistryKey regKey = Registry.LocalMachine;
         regKey = regKey.CreateSubKey("SOFTWARE\\Data
                  Dynamics\\ActiveReports.NET\\SampleDB");
         return (string)regKey.GetValue("");
}
```

Adding code to the ReportStart event

To write the code in Visual Basic

- Right-click in any section of the design window of rptModifyDS, and click on **View Code** to display the code view for the report. At the top left of the code view for rptModifyDS, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for rptModifyDS's ReportStart event. Add code to the handler to:

    - Change the data source at run time

To write the code in C#

- Click in the gray area below rptModifyDS to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for rptModifyDS's ReportStart event. Add code to the handler to:

    - Change the data source at run time

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_cnn As OleDbConnection
Dim m_dbpath As String
Dim m_cnnString As String
Dim sqlString As String
Private Sub rptModifyDS_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart
         m_dbpath = getDatabasePath()
         m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;_
                  Data Source=" + m_dbpath + "\\NWind.MDB"
         sqlString = "SELECT * from products"
         m_cnn = New OleDbConnection(m_cnnString)
         Dim m_cmd As New OleDbCommand(sqlString, m_cnn)
         If m_cnn.State = ConnectionState.Closed Then
                  m_cnn.Open()
         End If
End Sub

[C#]

private static OleDbConnection m_cnn;
private static OleDbDataReader  m_reader;
private void rptModifyDS_ReportStart(object sender, System.EventArgs
         eArgs)
{
         string m_dbPath = getDatabasePath();
         string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;
                  Data Source=" + m_dbPath + "\\NWind.MDB";
         string sqlString = "SELECT * from products";
         m_cnn = new OleDbConnection(m_cnnString);
         OleDbCommand m_Cmd = new OleDbCommand(sqlString,m_cnn);
         if(m_cnn.State == ConnectionState.Closed)
         {
                  m_cnn.Open();
         }
}
```

Saving and Loading to a Memory Stream

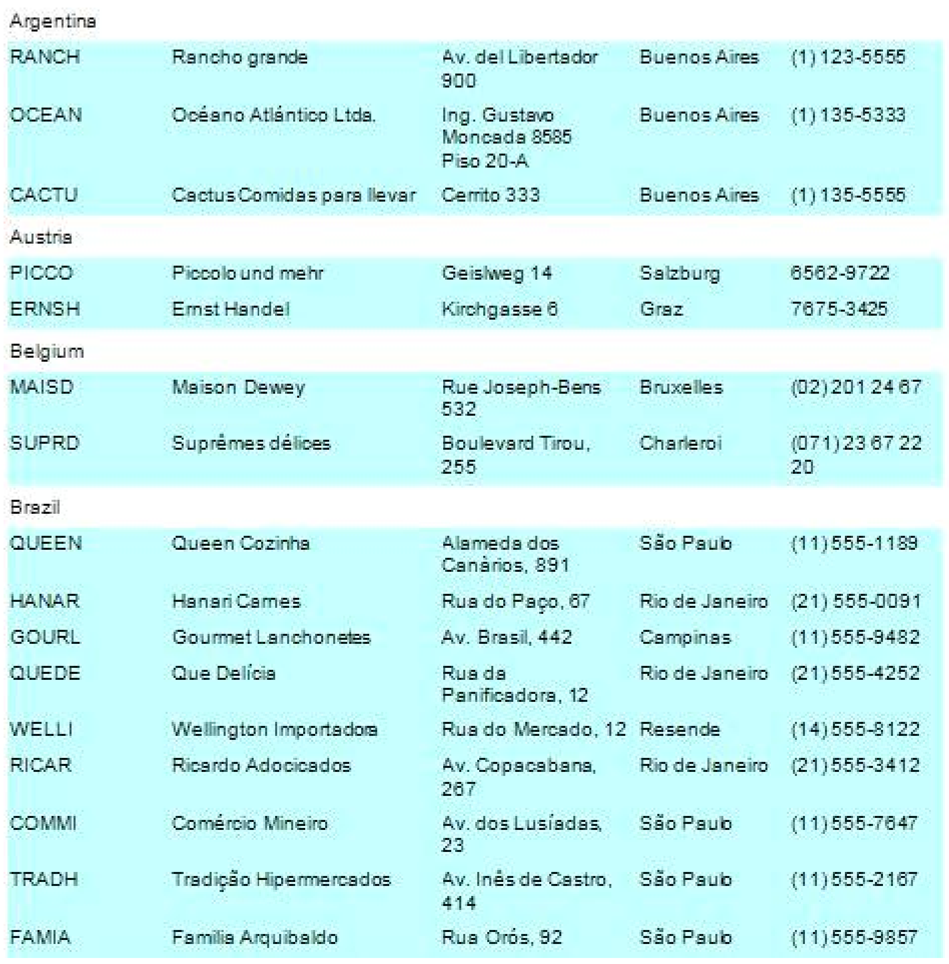
ActiveReports allows you to save and load a report as a memory stream. Saving a report as a memory stream makes it possible to save and load reports from a database or pass reports back and forth between DLLs.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding the viewer to Form1
- Adding code to the Form_Load event to save the report to a memory stream and load it to the viewer

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Argentina				
RANCH	Rancho grande	Av. del Libertador 900	Buenos Aires	(1) 123-5555
OCEAN	Océano Atlántico Ltda.	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	(1) 135-5333
CACTU	Cactus Comidas para llevar	Cerrito 333	Buenos Aires	(1) 135-5555
Austria				
PICCO	Piccolo und mehr	Geislweg 14	Salzburg	6562-9722
ERNSH	Ernst Handel	Kirchgasse 6	Graz	7675-3425
Belgium				
MAISD	Maison Dewey	Rue Joseph-Bens 532	Bruxelles	(02) 201 24 67
SUPRD	Suprêmes délices	Boulevard Tirou, 255	Charleroi	(071) 23 67 22 20
Brazil				
QUEEN	Queen Cozinha	Alameda dos Canàrios, 891	São Paulo	(11) 555-1189
HANAR	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	(21) 555-0091
GOURL	Gourmet Lanchonetes	Av. Brasil, 442	Campinas	(11) 555-9482
QUEDE	Que Delícia	Rua da Panificadora, 12	Rio de Janeiro	(21) 555-4252
WELLI	Wellington Importadora	Rua do Mercado, 12	Resende	(14) 555-8122
RICAR	Ricardo Adocicados	Av. Copacabana, 267	Rio de Janeiro	(21) 555-3412
COMMI	Comércio Mineiro	Av. dos Lusíadas, 23	São Paulo	(11) 555-7647
TRADH	Tradição Hipermercados	Av. Inês de Castro, 414	São Paulo	(11) 555-2167
FAMIA	Familia Arquibaldo	Rua Orós, 92	São Paulo	(11) 555-9857

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMemoryStream.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers ORDER BY country".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptMemoryStream.
2. Make the following changes to the group header:
   - o Change the name to ghCustomers
   - o Change the DataField property to Country
3. Add the following controls to the rptMemoryStream:

Control	DataField	Name	Text/Caption	Section	Location
TextBox	Country	**txtCountry**	Country	GroupHeader	0, 0
TextBox	CustomerID	**txtCustomerID**	Customer ID	Detail	0, 0
TextBox	CompanyName	**txtCompanyName**	Company Name	Detail	1.125, 0
TextBox	Address	**txtAddress**	Address	Detail	3.01, 0
TextBox	City	**txtCity**	City	Detail	4.375, 0
TextBox	Phone	**txtPhone**	Phone	Detail	5.437, 0

Adding the viewer to Form1

To add the viewer to Form1

1. Click on the ActiveReports viewer control in the appropriate toolbox and drag it onto Form1.
2. Set the viewer control's Dock property to Fill.

Adding code to the Form1_Load event

To write the code in Visual Basic

- Right-click in any section of Form1, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for Form1, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

  - Save the report to a memory stream
  - Load the memory stream to the ActiveReports viewer

To write the code in C#

- Click at the top of Form1 to select the Windows Form. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Load*. This creates an event-handling method for the Form1_Load event. Add code to the handler to:

  - Save the report to a memory stream
  - Load the memory stream to the ActiveReports viewer

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Form1_Load(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles MyBase.Load
        Dim strm As New System.IO.MemoryStream()
        Dim rpt As New rptMemoryStream()

        rpt.Run()
        rpt.Document.Save(strm)
        Dim theBytes(strm.Length) As Byte
        strm.Read(theBytes, 0, Int(strm.Length))
        strm.Position = 0
        Viewer1.Document.Load(strm)
    End Sub

[C#]

private void Form1_Load(object sender, System.EventArgs e)
{
         System.IO.MemoryStream strm = new System.IO.MemoryStream();
         rptMemoryStream rpt = new rptMemoryStream();
         rpt.Run();
         rpt.Document.Save(strm);

         byte[] theBytes = new byte[strm.Length];
         strm.Read(theBytes, 0, (int)strm.Length);

         strm.Position =0;
         viewer1.Document.Load(strm);
}
```

Scripting Walkthroughs

ActiveReports allows you to use scripting to provide ease in reporting functionality. Scripting permits reports saved to an RPX file to contain code. This characteristic allows the options of stand-alone reporting and web reporting without requiring .vb or .cs files. By including scripting when the report is saved as an RPX file, it can later by loaded, run and displayed directly to the viewer control without using the designer. Scripting can also be used in conjunction with RPX files to allow distributed reports to be updated without recompiling.

- Scripting and Subreports
- Scripting and Simple Reports

Scripting and Simple Reports

ActiveReports allows you to use scripting to permit reports saved to an XML file to contain code. By including scripting when the RPX files are saved into XML, the reports later can be loaded, run and displayed directly to the viewer control without needing to use the designer.

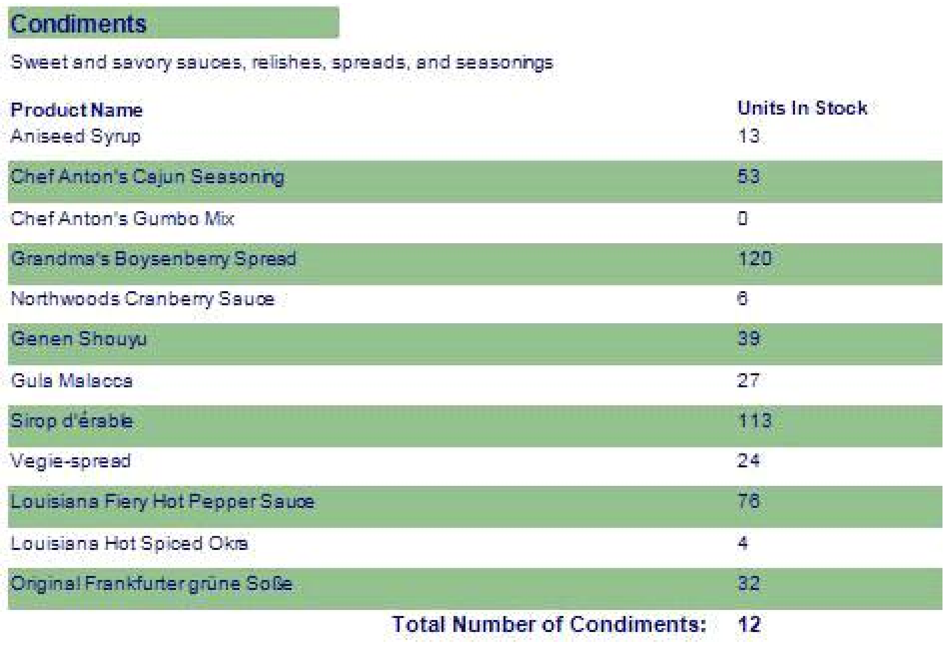
This walkthrough illustrates how to include scripting in a report.

This walkthrough is split into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Adding controls to contain the data
- Adding scripting to set the data connection, add fields to the report's fields collection and populate the report fields

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

Product Name	Units In Stock
Aniseed Syrup	13
Chef Anton's Cajun Seasoning	53
Chef Anton's Gumbo Mix	0
Grandma's Boysenberry Spread	120
Northwoods Cranberry Sauce	6
Genen Shouyu	39
Gula Malacca	27
Sirop d'érable	113
Vegie-spread	24
Louisiana Fiery Hot Pepper Sauce	76
Louisiana Hot Spiced Okra	4
Original Frankfurter grüne Soße	32
Total Number of Condiments:	12

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptScript.
4. Click **Open**.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to your report.
2. Make the following changes to the group header:
   - o Change the name to ghCategories
   - o Change the DataField property to CategoryID
   - o Set the GroupKeepTogether property to All
   - o Set the KeepTogether property to True
3. Make the following change to the group footer:
   - o Change the name to gfCategories
4. Add the following controls to the GroupHeader section:

Control	DataField	Name	Text/Caption	Location
TextBox	CategoryName	**txtCategoryName**	Category Name	0, 0

TextBox	Description	**txtDescription**	Description	0, 0.25
Label	(Empty string)	**lblProductName**	Product Name	0, 0.572
Label	(Empty string)	**lblUnitsInStock**	Units In Stock	5.0625, 0.562

5. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	Product Name	0, 0
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	5.0625, 0

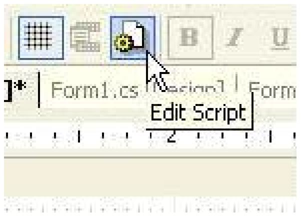
6. Add the following controls to the GroupFooter section:

Control	DataField	Name	Text/Caption	Misc Details	Location
Label	TotalLabel	**lblTotalLabel**	Total Label	(Empty string)	2.25, 0
TextBox	ProductName	**txtTotalItems**	Total Items	Summary Type = Subtotal SummaryFunc = Count SummaryRunning = Group SummaryGroup = ghCategories	5.0625, 0

Adding scripting to add data to the controls

To add scripting to the report

1. Click on the Edit Script icon on the report toolbar.



2. Add the following scripting code.

The following example shows what the scripting code looks like:

```
[C#]

private static System.Data.OleDb.OleDbConnection m_cnn;
private static System.Data.OleDb.OleDbDataReader  m_reader;

private string getDatabasePath()
{
         Microsoft.Win32.RegistryKey regKey =
                  Microsoft.Win32.Registry.LocalMachine;

         regKey = regKey.CreateSubKey("SOFTWARE\\Data
                Dynamics\\ActiveReports.NET\\SampleDB");

         return (string)regKey.GetValue("");
}
public void ActiveReport_ReportStart()
{
         string m_dbPath = getDatabasePath();
```

```
	string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
		Source=" + m_dbPath + "\\NWind.MDB";
	string sqlString = "SELECT * FROM categories INNER JOIN products
		ON categories.categoryid =
		products.categoryid ORDER BY products.categoryid,
		products.productid";
	m_cnn = new System.Data.OleDb.OleDbConnection(m_cnnString);
	System.Data.OleDb.OleDbCommand m_Cmd = new
		System.Data.OleDb.OleDbCommand(sqlString,m_cnn);
	if(m_cnn.State == System.Data.ConnectionState.Closed)
	{
		m_cnn.Open();
	}
	m_reader = m_Cmd.ExecuteReader();
}

public void ActiveReport_DataInitialize()
{
	rpt.Fields.Add("CategoryID");
	rpt.Fields.Add("CategoryName");
	rpt.Fields.Add("ProductName");
	rpt.Fields.Add("UnitsInStock");
	rpt.Fields.Add("Description");
	rpt.Fields.Add("TotalLabel");
}

public bool ActiveReport_FetchData(bool eof)
{
	try
	{
		m_reader.Read();
		rpt.Fields["CategoryID"].Value = m_reader
			["categories.CategoryID"].ToString();
		rpt.Fields["CategoryName"].Value = m_reader
			["CategoryName"].ToString();
		rpt.Fields["ProductName"].Value = m_reader
			["ProductName"].ToString();
		rpt.Fields["UnitsInStock"].Value = m_reader
			["UnitsInStock"].ToString();
		rpt.Fields["Description"].Value = m_reader
			["Description"].ToString();
		rpt.Fields["TotalLabel"].Value = "Total Number of
			" + m_reader["CategoryName"].
			ToString() + ":";
		eof = false;
	}
	catch
	{
		eof = true;
	}

	return eof;
}
bool m_color;
public void Detail_Format()
{
	if(m_color)
	{
		m_color =false;
		rpt.Sections["Detail"].BackColor =
			System.Drawing.Color.DarkSeaGreen;
	}
	else
	{
		rpt.Sections["Detail"].BackColor =
			System.Drawing.Color.Transparent;
		m_color = true;
	}
}
```

3. Click **OK** to continue.

Scripting and Subreports

ActiveReports allows you to use scripting to permit reports saved to an XML file to contain code. By including scripting when the RPX files are saved into XML, the reports later can be loaded, run and displayed directly to the viewer control without needing to use the designer.

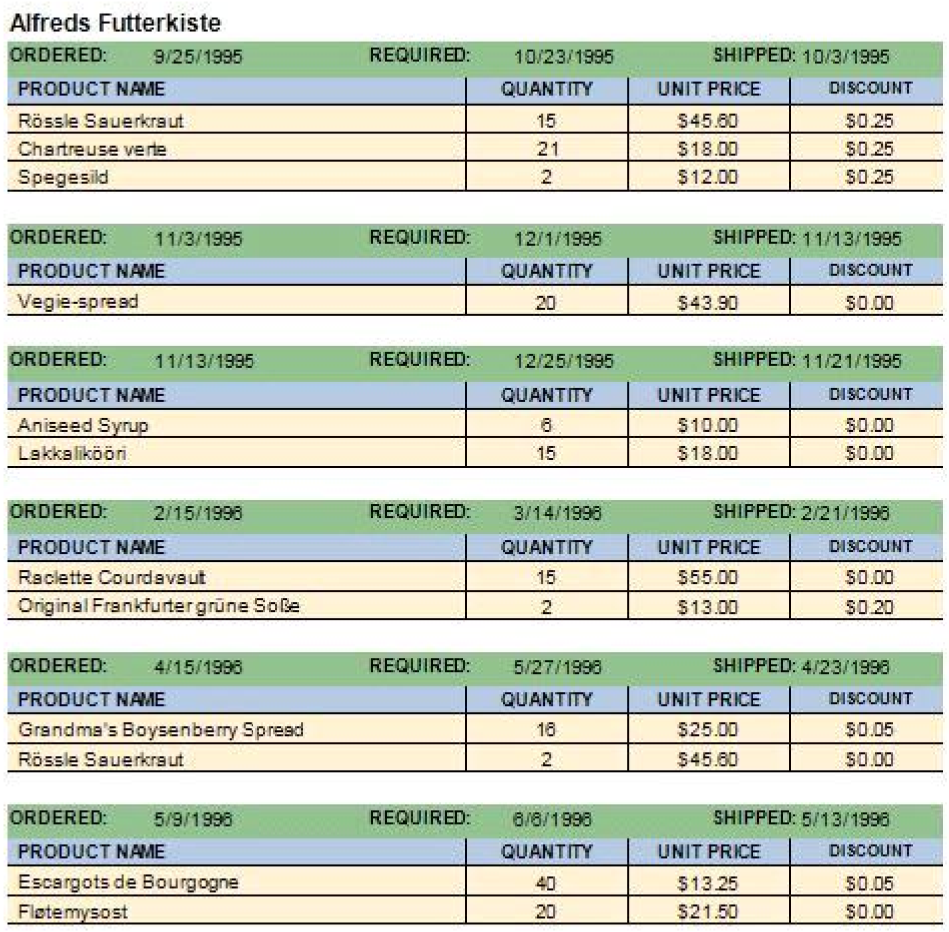
This walkthrough illustrates how to use scripting when creating a subreport.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the subreport to a data source
- Adding controls to each report to display the data
- Adding the scripting code for rptMain
- Adding the scripting code for rptSub

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



Alfreds Futterkiste

ORDERED: 9/25/1995 REQUIRED: 10/23/1995 SHIPPED: 10/3/1995

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Rössle Sauerkraut	15	$45.60	$0.25
Chartreuse verte	21	$18.00	$0.25
Spegesild	2	$12.00	$0.25

ORDERED: 11/3/1995 REQUIRED: 12/1/1995 SHIPPED: 11/13/1995

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Vegie-spread	20	$43.90	$0.00

ORDERED: 11/13/1995 REQUIRED: 12/25/1995 SHIPPED: 11/21/1995

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Aniseed Syrup	6	$10.00	$0.00
Lakkalikööri	15	$18.00	$0.00

ORDERED: 2/15/1996 REQUIRED: 3/14/1996 SHIPPED: 2/21/1996

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Raclette Courdavault	15	$55.00	$0.00
Original Frankfurter grüne Soße	2	$13.00	$0.20

ORDERED: 4/15/1996 REQUIRED: 5/27/1996 SHIPPED: 4/23/1996

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Grandma's Boysenberry Spread	16	$25.00	$0.05
Rössle Sauerkraut	2	$45.60	$0.00

ORDERED: 5/9/1996 REQUIRED: 6/6/1996 SHIPPED: 5/13/1996

PRODUCT NAME	QUANTITY	UNIT PRICE	DISCOUNT
Escargots de Bourgogne	40	$13.25	$0.05
Fløtemysost	20	$21.50	$0.00

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMain.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptSub.
7. Click **Open**.

Connecting the subreport to a data source

To connect the subreport to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from [order details] inner join products on [order details].productid = products.productid where [order details].orderid = <%orderID%>".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptMain.
2. Make the following changes to the group header:
   - o Change the name to ghCompanies
   - o Change the DataField to CompanyName
   - o Change the GroupKeepTogether property to All
3. Add a second GroupHeader/Footer section to rptMain.
4. Make the following changes to the group header:
   - o Change the name to ghOrders
   - o Change the DataField to OrderDate
   - o Change the GroupKeepTogether property to All
5. Add the following controls to rptMain, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	ghCompanies	0, 0
TextBox	OrderDate	**txtOrderDate**	Order Date	ghOrders	1, 0
TextBox	RequiredDate	**txtRequiredDate**	Required Date	ghOrders	3.5, 0
TextBox	ShippedDate	**txtShippedDate**	Shipped Date	ghOrders	5.5, 0
Label	(Empty string)	**lblOrderDate**	Ordered:	ghOrders	0, 0
Label	(Empty string)	**lblRequiredDate**	Required:	ghOrders	2.5, 0
Label	(Empty string)	**lblShippedDate**	Shipped:	ghOrders	4.875, 0
Subreport	(Empty string)	**Subreport1**	(Empty string)	Detail	0, 0

6. Add a GroupHeader/Footer section to rptSub.
7. Make the following changes to the group header:
   - o Change the name to ghOrderDetails
   - o Change the DataField to [order details].orderID

8. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblUnitPrice**	Unit Price	4.375, 0
Label	**lblDiscount**	Discount	5.5, 0
Label	**lblProductName**	Product Name	0.0625, 0
Label	**lblQuantity**	Quantity	3.25, 0

Line	**Line1**	(Empty string)	X1 = 3.1875
			Y1 = 0
			X2 = 3.1875
			Y2 = 0.1875

Line	**Line2**	(Empty string)	X1 = 4.3125
			Y1 = 0
			X2 = 4.3125
			Y2 = 0.1875

Line	**Line3**	(Empty string)	X1 = 5.4375
			Y1 = 0
			X2 = 5.4375
			Y2 = 0.1875

Line	**Line4**	(Empty string)	X1 = 0
			Y1 = 0.1875
			X2 = 6.5
			Y2 = 0.1875

9. Add the following controls to the Detail section:

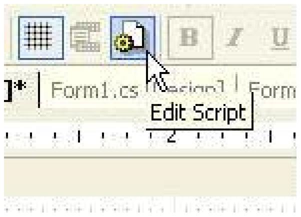
Control	DataField	Name	Text/Caption	Misc Details	Location
TextBox	ProductName	**txtProductName**	Product Name	(Empty string)	0.0625, 0
TextBox	Quantity	**txtQuantity**	Quantity	(Empty string)	3.25, 0
TextBox	order details.UnitPrice	**txtUnitPrice**	Unit Price	OutputFormat = Currency	4.375, 0
TextBox	Discount	**txtDiscount**	Discount	OutputFormat = Currency	5.5, 0
Line	(Empty string)	**Line5**	(Empty string)	(Empty string)	X1 = 3.1875 Y1 = 0 X2 = 3.1875 Y2 = 0.1875
Line	(Empty string)	**Line6**	(Empty string)	(Empty string)	X1 = 4.3125 Y1 = 0 X2 = 4.3125 Y2 = 0.1875
Line	(Empty string)	**Line7**	(Empty string)	(Empty string)	X1 = 5.4375 Y1 = 0 X2 = 5.4375 Y2 = 0.1875

Line	(Empty string)	**Line8**	(Empty string)	(Empty string)	X1 = 0 Y1 = 0.1875 X2 = 6.5 Y2 = 0.1875

Adding the scripting code for rptMain

To add scripting to the report

1. Click on the Edit Script icon on the report toolbar.



2. Add the following scripting code.

The following example shows what the scripting code looks like:

```
[C#]

private string getDatabasePath()
{
         Microsoft.Win32.RegistryKey regKey =
                  Microsoft.Win32.Registry.LocalMachine;
         regKey = regKey.CreateSubKey("SOFTWARE\\Data
                  Dynamics\\ActiveReports.NET\\SampleDB");
         return (string)regKey.GetValue("");
}
public void ActiveReport_ReportStart()
{
         string m_dbPath = getDatabasePath();
         string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
                  Source=" + m_dbPath + "\\NWind.MDB";
         string sqlString = "Select * from orders inner join customers on
                  orders.customerid = customers.customerid order by
                  CompanyName, OrderDate";
         DataDynamics.ActiveReports.DataSources.
                  OleDBDataSource m_ds = new
                  DataDynamics.ActiveReports.
                  DataSources.OleDBDataSource();
         m_ds.ConnectionString = m_cnnString;
         m_ds.SQL = sqlString;
         rpt.DataSource = m_ds;
}
public void Detail_Format()
{
         DataDynamics.ActiveReports.ActiveReport rptSub = new
                  DataDynamics.ActiveReports.ActiveReport();
         rptSub.LoadLayout(System.Windows.Forms.Application.StartupPath +
                  @"\RPX Files\Orders\" +
                  ((SubReport)rpt.Sections["Detail"].Controls["SubReport1"]).
                  ReportName);
         ((SubReport)rpt.Sections["Detail"].Controls["SubReport1"]).
                  Report = rptSub;
}
```

3. Click **OK** to continue.

Adding the scripting code for rptSub

To add scripting to the report

1. Click on the Edit Script icon on the report toolbar.
2. Add the following scripting code.

The following example shows what the scripting code looks like:

```
[C#]

int cnt;
public void Detail_Format()
{
         cnt++;
         if(cnt % 2 == 0)
         {
                  rpt.Sections["Detail"].BackColor =
                           System.Drawing.Color.White;
         }
         else
         {
                  rpt.Sections["Detail"].BackColor =
                           System.Drawing.Color.BlanchedAlmond;
         }
}
```

3. Click **OK** to continue.

Style Sheets

ActiveReports adds style class names to allow controls to be formatted easily. With the use of style sheets, groups of controls can be set to a single style with just a few clicks. *ClassName* and the different control's style property can also be used to create specialized styles in code and through scripting.

This walkthrough illustrates how to create and use style sheets in a report.

This walkthrough is split up into the following activities:

- Opening an existing ActiveReport
- Creating style sheets
- Using created style sheets in your report at design time
- Using created style sheets in your report at run time

Opening an existing ActiveReport

To open an existing ActiveReport

1. Click **Open** > **Project**.
2. Select your ActiveReport project from the appropriate location and click on it to select it.
3. Click **Open**.

Creating Style Sheets

To create a style sheet

1. Click anywhere on the report design surface to select it.
2. Click on **Report > Settings...**
3. Click on *Styles* to display the style sheet.
4. Select **New** to add a new style, or select a predefined style to modify.
5. Name the new style "MyNewStyle" and select a base style.
6. Modify the properties to set up the desired effect and click **OK**.

Using Created Style Sheets in a Report at Design Time

To use a created style sheet at design time

1. Click on the control to which you wish to apply the style.
2. Select "MyNewStyle" from the style sheets drop-down box.
3. The new style is applied to your selected control.

Using Created Style Sheets in a Report at Run Time

To write the code in Visual Basic

- Right-click in any section of the design window of your report, and click on **View Code** to display the code view for the report. At the top left of the code view for the report, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for the report's Detail_Format event. Add code to the handler to:

    - Update the selected control with the style sheet chosen

To write the code in C#

- Click in the Detail section of your report to select it. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for the report's Detail_Format event. Add code to the handler to:

    - Update the selected control with the style sheet chosen

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles Detail.Format
        Me.TextBox1.ClassName = "MyNewStyle"
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         this.TextBox1.ClassName = "MyNewStyle";
}
```

Subreports Walkthroughs

With ActiveReports, reports may contain any number of child reports by using the Subreport control. Child reports, or subreports, are executed each time the parent section (i.e. the section in which the Subreport control is placed) is printed.

- o Hierarchical Subreports
- o Nested Subreports
- o Simple Subreports

Hierarchical Subreports

ActiveReports allows reports to contain any number of child reports by using the Subreport control. Child reports, or subreports, are executed each time the parent section (i.e. the section in which the Subreport control is placed) is printed.

Note Subreports will not render PageHeader/Footer sections.

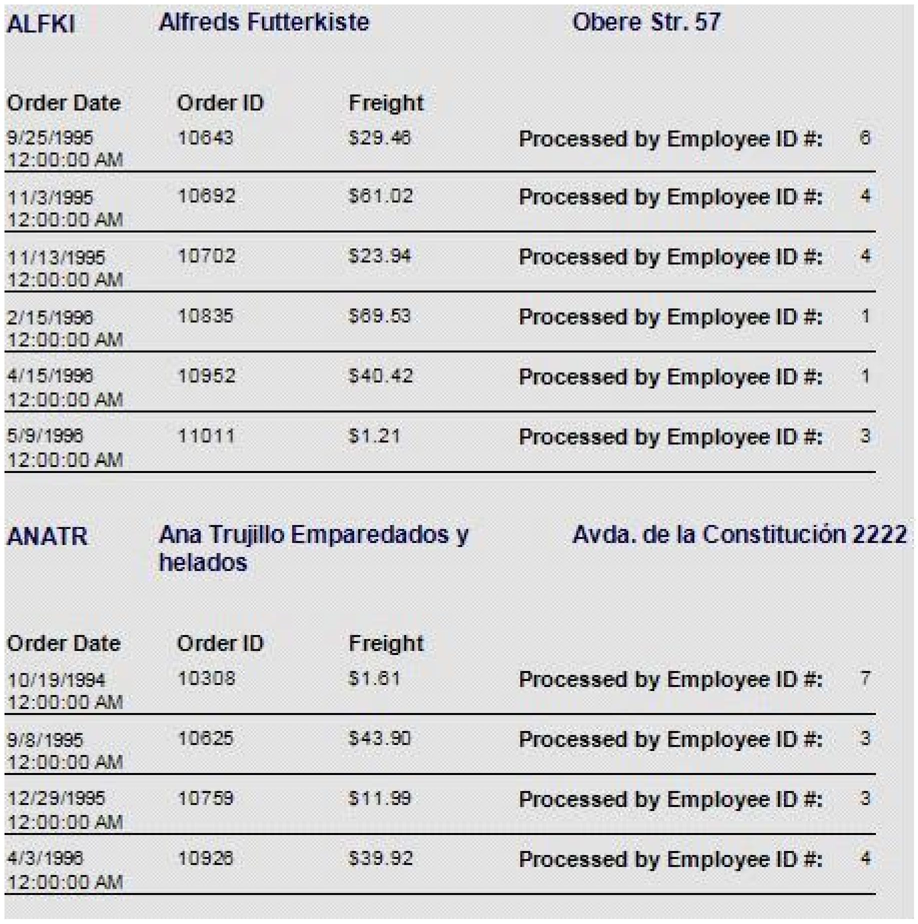
This walkthrough illustrates how to set up a bound subreport by setting the Subreport control's Report property to the child report and how to modify the subreport record source from the data in the parent report to retrieve the correct information.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting rptCustomers to a data source
- Adding controls to each report to display the data
- Adding the code needed to set the subreport control equal to rptOrders

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



ALFKI	Alfreds Futterkiste		Obere Str. 57	
Order Date	Order ID	Freight		
9/25/1995 12:00:00 AM	10643	$29.46	Processed by Employee ID #:	6
11/3/1995 12:00:00 AM	10692	$61.02	Processed by Employee ID #:	4
11/13/1995 12:00:00 AM	10702	$23.94	Processed by Employee ID #:	4
2/15/1996 12:00:00 AM	10835	$69.53	Processed by Employee ID #:	1
4/15/1996 12:00:00 AM	10952	$40.42	Processed by Employee ID #:	1
5/9/1996 12:00:00 AM	11011	$1.21	Processed by Employee ID #:	3

ANATR	Ana Trujillo Emparedados y helados		Avda. de la Constitución 2222	
Order Date	Order ID	Freight		
10/19/1994 12:00:00 AM	10308	$1.61	Processed by Employee ID #:	7
9/8/1995 12:00:00 AM	10625	$43.90	Processed by Employee ID #:	3
12/29/1995 12:00:00 AM	10759	$11.99	Processed by Employee ID #:	3
4/3/1996 12:00:00 AM	10926	$39.92	Processed by Employee ID #:	4

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptCustomers.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptOrders.
7. Click **Open**.

Connecting rptCustomers to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.

6. In the Query field, type "SHAPE {SELECT CustomerID, CompanyName, Address FROM Customers} APPEND ({SELECT CustomerID, OrderID, Freight, OrderDate, EmployeeID FROM Orders} AS CustomerOrders RELATE CustomerID TO CustomerID)".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add the following controls to rptCustomers, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	CustomerID	**txtCustomerID**	Customer ID	0, 0
TextBox	CompanyName	**txtCompanyName**	Company Name	1.0625, 0
TextBox	Address	**txtAddress**	Address	3.9375, 0
Subreport	CustomerOrders	**Subreport1**	(Empty string)	0, 0.5

2. Add a GroupHeader/Footer section to rptOrders
3. Make the following changes to the group header:
   - o Change the name to ghOrders
   - o Change the DataField property to CustomerID
4. Add the following controls to rptOrders, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
Label	(Empty string)	**lblOrderDate**	Order Date	GroupHeader	0, 0
Label	(Empty string)	**lblOrderID**	Order ID	GroupHeader	1.1875, 0
Label	(Empty string)	**lblFreight**	Freight	GroupHeader	2.375, 0
TextBox	OrderDate	**txtOrderDate**	Order Date	Detail	0, 0
TextBox	OrderID	**txtOrderID**	Order ID	Detail	1.1875, 0
TextBox	Freight	**txtFreight**	Freight	Detail	2.375, 0
TextBox	EmployeeID	**txtEmployeeID**	Employee ID	Detail	5.9375, 0
Label	(Empty string)	**lblProcessed**	Processed by Employee ID #:	Detail	3.5625, 0
Line	(Empty string)	**Line1**	(Empty string)	Detail	X1: 0 Y1: 0.25 X2: 6.5 Y2: 0.25

Adding the code needed to set the subreport control equal to rptOrders

To write the code in Visual Basic

- Right-click in any section of the design window of rptCustomers, and click on **View Code** to display the code view for the report. At the top left of the code view for rptCustomers, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptCustomers' Detail_Format event. Add code to the handler to:

- Set the subreport control equal to rptOrders

To write the code in C#

- Click in the Detail section of rptCustomers to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptCustomers' Detail_Format event. Add code to the handler to:
    - Set the subreport control equal to rptOrders

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal e _
         As System.EventArgs) Handles Detail.Format
        Me.SubReport1.Report = New rptOrders()
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         this.SubReport1.Report = new rptOrders();
}
```

Nested Subreports

When setting up embedded subreports in ActiveReports, the principles are the same as when setting up simple subreports but are applied to the child-grandchild reports.

Note Subreports will not render PageHeader/Footer sections.

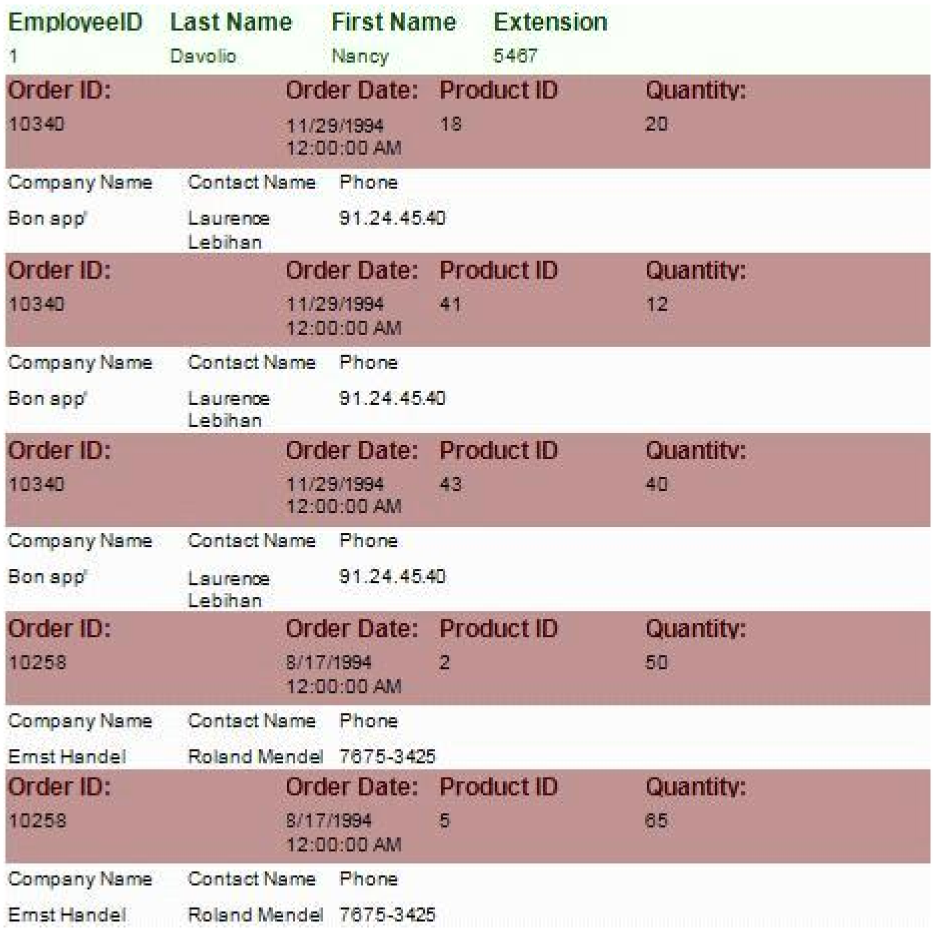
This walkthrough illustrates how to set up embedded subreports.

This walkthrough is split up into the following activities:

- Adding three ActiveReports to a Visual Studio project
- Connecting each report to a data source
- Adding controls to each report to display the data
- Adding the code needed to set the subreport controls equal to their corresponding reports

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



```
EmployeeID  Last Name  First Name  Extension
1           Davolio    Nancy       5467
Order ID:                 Order Date:   Product ID   Quantity:
10340                     11/29/1994    18           20
                          12:00:00 AM
Company Name   Contact Name   Phone
Bon app'       Laurence       91.24.45.40
               Lebihan
Order ID:                 Order Date:   Product ID   Quantity:
10340                     11/29/1994    41           12
                          12:00:00 AM
Company Name   Contact Name   Phone
Bon app'       Laurence       91.24.45.40
               Lebihan
Order ID:                 Order Date:   Product ID   Quantity:
10340                     11/29/1994    43           40
                          12:00:00 AM
Company Name   Contact Name   Phone
Bon app'       Laurence       91.24.45.40
               Lebihan
Order ID:                 Order Date:   Product ID   Quantity:
10258                     8/17/1994     2            50
                          12:00:00 AM
Company Name   Contact Name   Phone
Ernst Handel   Roland Mendel  7675-3425
Order ID:                 Order Date:   Product ID   Quantity:
10258                     8/17/1994     5            65
                          12:00:00 AM
Company Name   Contact Name   Phone
Ernst Handel   Roland Mendel  7675-3425
```

Adding three ActiveReports to a Visual Studio project

To add three ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptEmployees.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptOrders.
7. Click **Open**.
8. Click on **Project > Add New Item**.
9. Select **ActiveReports file** and rename the file rptCustomers.
10. Click **Open**.

Connecting rptEmployees to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**

4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from employees".
7. Click **OK** to return to the report design surface.

Connecting rptOrders to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from orders inner join [order details] on orders.orderID = [order details].orderID where orders.employeeID = <%employeeID%>".
7. Click **OK** to return to the report design surface.

Connecting rptCustomers to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from customers where customerID = '<%CustomerID%>' ".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add a GroupHeader/Footer section to rptEmployees
2. Make the following changes to the group header:
   - Change the name to ghEmployees
   - Change the DataField property to EmployeeID
3. Add the following controls to rptEmployees, naming them as indicated:

Control	DataField	Name	Text/Caption	Section	Location
TextBox	EmployeeID	**txtEmployeeID**	Employee ID	GroupHeader	0, 0
TextBox	Extension	**txtExtension**	Extension	GroupHeader	3.375, 0
TextBox	LastName	**txtLastName**	Last Name	GroupHeader	1.125, 0
TextBox	FirstName	**txtFirstName**	First Name	GroupHeader	2.25, 0
Label	(Empty string)	**lblEmployeeID**	Employee ID	GroupHeader	0, 0

Label	(Empty string)	**lblExtension**	Extension	GroupHeader	3.375, 0
Label	(Empty string)	**lblLastName**	Last Name	GroupHeader	1.125, 0
Label	(Empty string)	**lblFirstName**	First Name	GroupHeader	2.25, 0
Subreport	(Empty string)	**subOrders**	(Empty string)	Detail	0, 0

4. Add the following controls to the Detail section of rptOrders, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	OrderDate	**txtOrderDate**	Order Date	1.9375, 0.25
TextBox	Quantity	**txtQuantity**	Quantity	4.4375, 0.25
TextBox	orders.OrderID	**txtOrderID**	Order ID	0, 0.25
TextBox	ProductID	**txtProductID**	Product ID	3, 0.25
Label	(Empty string)	**lblOrderDate**	Order Date:	1.9375, 0
Label	(Empty string)	**lblQuantity**	Quantity:	4.4375, 0
Label	(Empty string)	**lblOrderID**	Order ID:	0, 0
Label	(Empty string)	**lblProductID**	Product ID:	3, 0
Subreport	(Empty string)	**subCustomers**	(Empty string)	0, 0.5

5. Add the following controls to the Detail section of rptCustomers, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	CompanyName	**txtCompanyName**	Company Name	0, 0.25
TextBox	ContactName	**txtContactName**	Contact Name	1.25, 0.25
TextBox	Phone	**txtPhone**	Phone	2.3125, 0.25
Label	(Empty string)	**lblCompanyName**	Company Name	0, 0
Label	(Empty string)	**lblContactName**	Contact Name	1.25, 0
Label	(Empty string)	**lblPhone**	Phone	2.3125, 0

Adding the code needed to set subOrders equal to rptOrders in rptEmployees

To write the code in Visual Basic

- Right-click in any section of the design window of rptEmployees, and click on **View Code** to display the code view for the report. At the top left of the code view for rptEmployees, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptEmployees' Detail_Format event. Add code to the handler to:
  - Set subOrders equal to rptOrders

To write the code in C#

- Click in the Detail section of rptEmployees to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptEmployees' Detail_Format event. Add code to the handler to:

    - Set subOrders equal to rptOrders

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles Detail.Format
        Dim rpt As New rptOrders()
        Me.subOrders.Report = rpt
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         rptOrders rpt = new rptOrders();
         subOrders.Report = rpt;
}
```

Adding the code needed to set subCustomers equal to rptCustomers in rptOrders

To write the code in Visual Basic

- Right-click in any section of the design window of rptOrders, and click on **View Code** to display the code view for the report. At the top left of the code view for rptOrders, click the drop-down arrow and select *Detail*. At the top right of the code window, click the drop-down arrow and select *Format*. This creates an event-handling method for rptOrders' Detail_Format event. Add code to the handler to:
    - Set subCustomers equal to rptCustomers

To write the code in C#

- Click in the Detail section of rptOrders to select the section. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Format*. This creates an event-handling method for rptOrders' Detail_Format event. Add code to the handler to:
    - Set subCustomers equal to rptCustomers

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub ghOrders_Format(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles ghOrders.Format
        Dim rpt As New rptCustomers()
        Me.subCustomers.Report = rpt
End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         rptCustomers rpt = new rptCustomers();
         subCustomers.Report = rpt;
}
```

Simple Subreports

ActiveReports allows reports to contain any number of child reports by using the Subreport control. Child reports, or subreports, are executed each time the parent section (i.e. the section in which the Subreport control is placed) is printed.

Note Subreports will not render PageHeader/Footer sections.

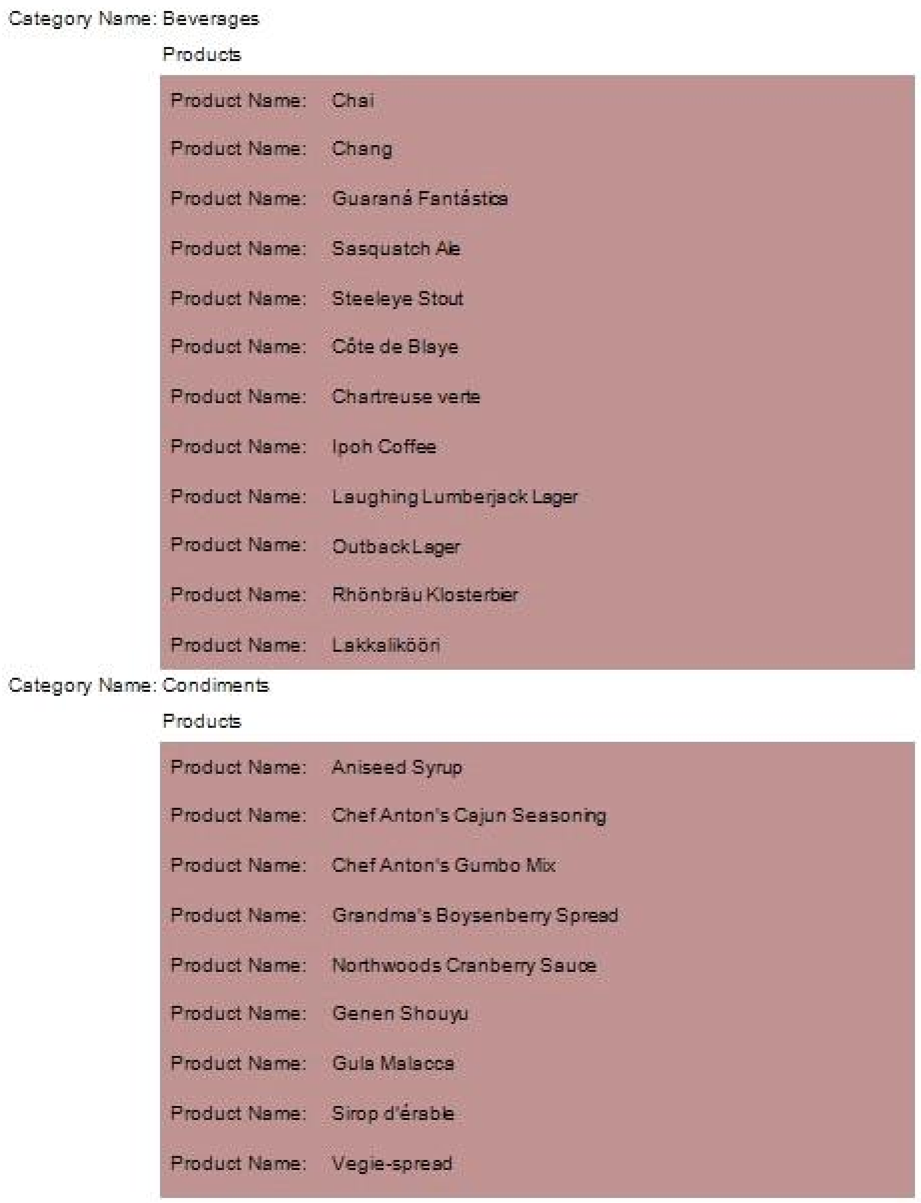
This walkthrough illustrates how to set up a bound subreport by setting the Subreport control's Report property to the child report and how to modify the subreport record source from the data in the parent report to retrieve the correct information.

This walkthrough is split up into the following activities:

- Adding two ActiveReports to a Visual Studio project
- Connecting the parent report to a data source
- Adding controls to display the data
- Adding the code needed to save the current record's CategoryID to use in the subreport's SQL query
- Adding the code to create a new data source, setting its connection string, setting its SQL query and setting the new data source equal to the subreport's data source

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have finished this walkthrough, you will have a report that looks similar to the following.



```
Category Name: Beverages
               Products
               Product Name:    Chai
               Product Name:    Chang
               Product Name:    Guaraná Fantástica
               Product Name:    Sasquatch Ale
               Product Name:    Steeleye Stout
               Product Name:    Côte de Blaye
               Product Name:    Chartreuse verte
               Product Name:    Ipoh Coffee
               Product Name:    Laughing Lumberjack Lager
               Product Name:    Outback Lager
               Product Name:    Rhönbräu Klosterbier
               Product Name:    Lakkalikööri
Category Name: Condiments
               Products
               Product Name:    Aniseed Syrup
               Product Name:    Chef Anton's Cajun Seasoning
               Product Name:    Chef Anton's Gumbo Mix
               Product Name:    Grandma's Boysenberry Spread
               Product Name:    Northwoods Cranberry Sauce
               Product Name:    Genen Shouyu
               Product Name:    Gula Malacca
               Product Name:    Sirop d'érable
               Product Name:    Vegie-spread
```

Adding two ActiveReports to a Visual Studio project

To add two ActiveReports to a Visual Studio project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptMain.
4. Click **Open**.
5. Click on **Project > Add New Item**.
6. Select **ActiveReports file** and rename the file rptSub.
7. Click **Open**.

Connecting the parent report to a data source

To connect the parent report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from categories".
7. Click **OK** to return to the report design surface.

Adding controls to display the data

To add controls to the reports

1. Add the following controls to the Detail section of rptMain, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
Label	(Empty string)	**lblProducts**	Products	1.0625, 0.25
Label	(Empty string)	**lblCategoryName**	Category Name:	0, 0
TextBox	CategoryName	**txtCategoryName**	CategoryName	1.06, 0
Subreport	(Empty string)	**ctlSubreport**	(Empty string)	1.0625, 0.5

2. Add the following controls to the Detail section of rptSub, naming them as indicated:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	ProductName	1.187, 0.06
Label	(Empty string)	**lblProductName**	Product Name:	0.06, 0.06

Adding the code needed to save the current record's categoryID

To write the code in Visual Basic

- Right-click in any section of the design window of rptMain, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMain, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptMain's FetchData event. Add code to the handler to:
    - Save the current record's categoryID to use in the subreport's SQL query

To write the code in C#

- Click in the gray area below rptMain to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptMain's FetchData event. Add code to the handler to:
    - Save the current record's categoryID to use in the subreport's SQL query

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_categoryID As String
Private Sub rptMain_FetchData(ByVal sender As Object, ByVal _
         eArgs As DataDynamics.ActiveReports._
         ActiveReport.FetchEventArgs) Handles MyBase.FetchData
        m_categoryID = Me.Fields("CategoryID").Value
End Sub

[C#]

string m_categoryID;
Private Void rptMain_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.
         FetchEventArgs eArgs)
{
         m_categoryID = Fields["CategoryID"].Value.ToString();
}
```

Adding the code to create a new data source

To write the code in Visual Basic

- Right-click in any section of the design surface of rptMain, and click on **View Code** to display the code view for the report. At the top left of the code view for rptMain, click the drop-down arrow and select *rptMain*. At the top right of the code window, click the drop-down arrow and select *Detail_Format*. This creates an event-handling method for the report's Detail_Format event. Add code to the handler to:

  - Create a new DataDynamics OleDBDataSource
  - Set the new data source's connection string
  - Set the new data source's SQL query
  - Set the subreport's data source equal to the new data source

To write the code in C#

- Click in the Detail section of rptMain to select the section. Click on the events icon in the **Properties** window to display available events for the Detail section. Double-click *Format*. This creates an event-handling method for rptMain's Detail_Format event. Add code to the handler to:

  - Create a new DataDynamics OleDBDataSource
  - Set the new data source's connection string
  - Set the new data source's SQL query
  - Set the subreport's data source equal to the new data source

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Detail_Format(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles Detail.Format
        Dim rpt As New rptSub()
        Dim subDS As New DataDynamics.ActiveReports. _
                   DataSources.OleDBDataSource()
        subDS.ConnectionString = Me.ds.ConnectionString
        subDS.SQL = "Select * from products where categoryID _
                   = " + m_categoryID
        rpt.DataSource = subDS
        Me.ctlSubreport.Report = rpt
 End Sub

[C#]

private void Detail_Format(object sender, System.EventArgs eArgs)
{
         rptSub rpt = new rptSub();
         DataDynamics.ActiveReports.DataSources.OleDBDataSource subDS =
                  new DataDynamics.ActiveReports.DataSources.
                  OleDBDataSource();
         subDS.ConnectionString = this.ds.ConnectionString;
         subDS.SQL = "Select * from products where categoryID =
                  " + m_categoryID;
         rpt.DataSource = subDS;
         ctlSubReport.Report = rpt;
}
```

Summary Fields

In ActiveReports, summary fields can be added to any section to calculate totals, counts, averages and other aggregations. The summary field's placement dictates when the section containing the field, and sections after it, will be printed. A section with a summary field will be delayed until all the calculations are completed. This allows summary fields to be placed ahead of the corresponding detail.

Summary fields are calculated according to the textbox's Summary properties. A summary textbox is updated with each new detail record. When a field is placed ahead of the Detail section (i.e. in the ReportHeader, PageHeader or GroupHeader sections), the Detail section is formatted with each record and the summary field is updated. When all records for the summary level are read, the header section is printed followed by the delayed sections.

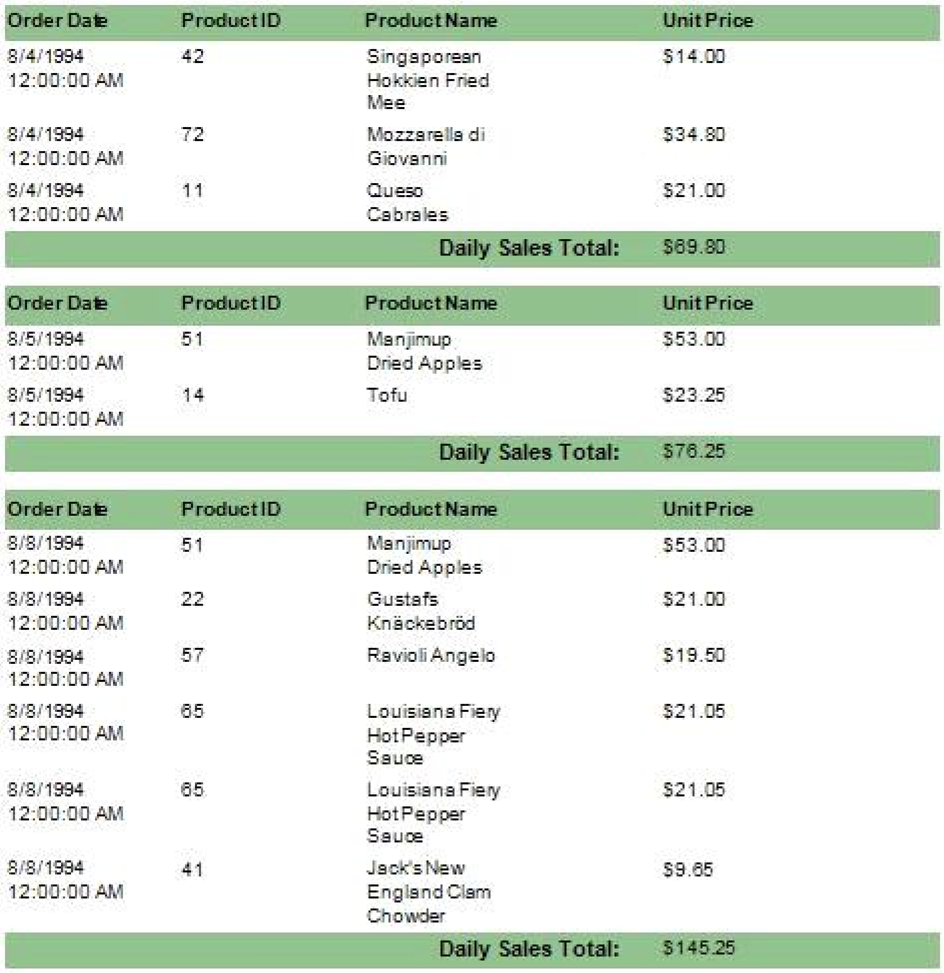
This walkthrough illustrates how to create a report with a summary field in the GroupFooter section.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Connecting the report to a data source
- Adding controls to the report to contain data

To complete the walkthrough, you must have access to the NorthWind database (Nwind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Order Date	Product ID	Product Name	Unit Price
8/4/1994 12:00:00 AM	42	Singaporean Hokkien Fried Mee	$14.00
8/4/1994 12:00:00 AM	72	Mozzarella di Giovanni	$34.80
8/4/1994 12:00:00 AM	11	Queso Cabrales	$21.00
		Daily Sales Total:	$69.80

Order Date	Product ID	Product Name	Unit Price
8/5/1994 12:00:00 AM	51	Manjimup Dried Apples	$53.00
8/5/1994 12:00:00 AM	14	Tofu	$23.25
		Daily Sales Total:	$76.25

Order Date	Product ID	Product Name	Unit Price
8/8/1994 12:00:00 AM	51	Manjimup Dried Apples	$53.00
8/8/1994 12:00:00 AM	22	Gustafs Knäckebröd	$21.00
8/8/1994 12:00:00 AM	57	Ravioli Angelo	$19.50
8/8/1994 12:00:00 AM	65	Louisiana Fiery Hot Pepper Sauce	$21.05
8/8/1994 12:00:00 AM	65	Louisiana Fiery Hot Pepper Sauce	$21.05
8/8/1994 12:00:00 AM	41	Jack's New England Clam Chowder	$9.65
		Daily Sales Total:	$145.25

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptSumFields.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.

6. In the Query field, type "SELECT DISTINCTROW Orders.*, [Order Details].*, Products.* FROM Products INNER JOIN (Orders INNER JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID) ON Products.ProductID = [Order Details].ProductID order by orderdate".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptSumFields.
2. Make the following changes to the group header:
   - o Change the name to ghOrders
   - o Change the DataField to OrderDate
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductID**	Product ID	1.208, 0
Label	**lblProductName**	Product Name	2.489, 0
Label	**lblUnitPrice**	Unit Price	4.56, 0
Label	**lblOrderDate**	Order Date	0, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	products.ProductID	txtProductID	Product ID	1.218, 0	(Empty string)
TextBox	ProductName	txtProductName	Product Name	2.5, 0	(Empty string)
TextBox	OrderDate	txtOrderDate	Order Date	0, 0	(Empty string)
TextBox	products.UnitPrice	txtUnitPrice	Unit Price	4.562, 0	Currency

5. Add the following controls to the GroupFooter section:

Control	DataField	Name	Text/Caption	Location	Misc Details
Label	(Empty string)	lblSalesTotal	Daily Sales Total	3, 0	(Empty string)
TextBox	products.UnitPrice	txtSalesTotal	Sales Total	4.562, 0	Output Format = Currency SummaryType = SubTotal SummaryRunning = Group SummaryGroup = ghOrders
Label	(Empty string)	lblWhiteLine	(Empty string)	0, 0.25	Background color = white

Note Distinct summarization can be used in a situation when the field's value repeats in several detail records and the summary function needs to include a single value from all repeating values. To do this, you would need to set the DistinctField property of the summary field to the appropriate value and set the SummaryFunc property to the appropriate distinct summary function.

Unbound Reports

ActiveReports gives you complete control to bind reports to any type of data source, including arrays, through its programmable object model. You can create a report without setting the report's data source then load the data from your data source into the report's control at run time.

The Fields property allows data binding between the control and the run-time fields. It also allows the control's DataField property to be set to any of the run-time defined names. The DataInitialize and FetchData events are used to define the run-time fields and feed the data values of these fields so they can be used with unbound controls.

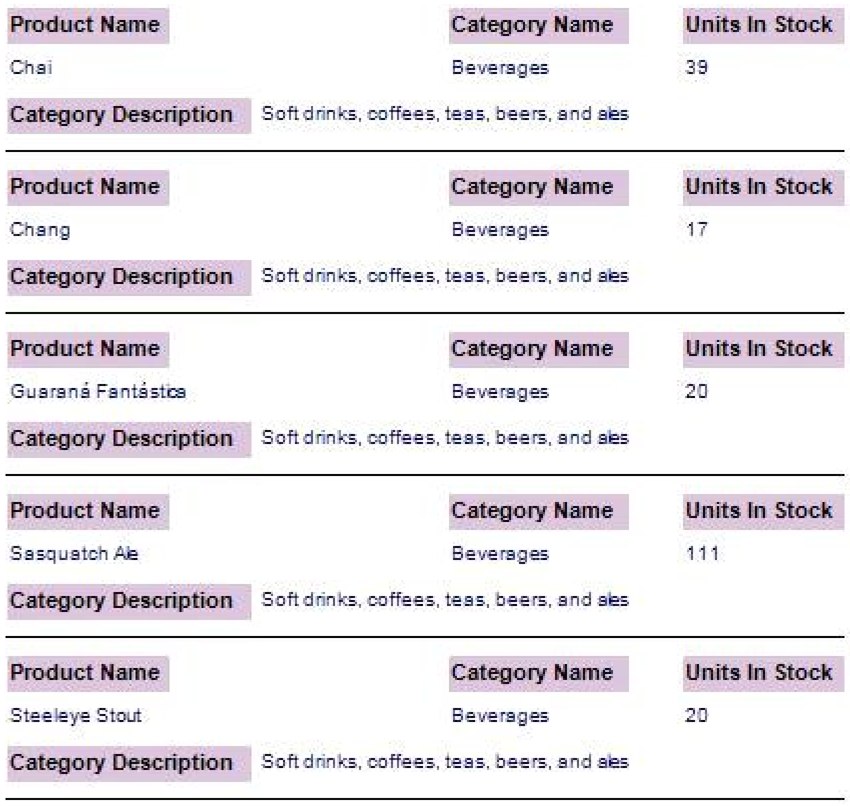
This walkthrough illustrates the fundamentals of using the DataInitialize and FetchData events to set up an unbound report.

This walkthrough is split into the following activities:

- Adding an ActiveReport to a Visual Studio project
- Adding code to connect the report to a data source
- Adding controls to contain the data
- Using the DataInitialize event to add fields to the report's fields collection
- Using the FetchData event to populate the report fields

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb).

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product Name Category Name Units In Stock
Chai Beverages 39
Category Description Soft drinks, coffees, teas, beers, and ales

Product Name Category Name Units In Stock
Chang Beverages 17
Category Description Soft drinks, coffees, teas, beers, and ales

Product Name Category Name Units In Stock
Guaraná Fantástica Beverages 20
Category Description Soft drinks, coffees, teas, beers, and ales

Product Name Category Name Units In Stock
Sasquatch Ale Beverages 111
Category Description Soft drinks, coffees, teas, beers, and ales

Product Name Category Name Units In Stock
Steeleye Stout Beverages 20
Category Description Soft drinks, coffees, teas, beers, and ales

Adding an ActiveReport to a Visual Studio project

To add an ActiveReport to your project

1. Open a new project in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptUnbound.
4. Click **Open**.

Adding code to connect the report to a data source

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnbound, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnbound, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *ReportStart*. This creates an event-handling method for rptUnbound's ReportStart event. Add code to the handler to:

  - Set the data source connection string
  - Set the data source SQL query
  - Open the connection to create the DataReader

To write the code in C#

- Click in the gray area below rptUnbound to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *ReportStart*. This creates an event-handling method for rptUnbound's ReportStart event. Add code to the handler to:

  - Set the data source connection string
  - Set the data source SQL query
  - Open the connection to create the DataReader

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim m_cnnString As String
Dim sqlString As String
Dim m_reader As OleDbDataReader
Dim m_cnn As OleDbConnection
Private Sub rptUnbound_ReportStart(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.ReportStart

        m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data _
                   Source=C:\Program Files\Data _
         Dynamics\ActiveReports.NET\Data\NWIND.MDB;_
                   Persist Security Info=False"
        sqlString = "SELECT * FROM categories INNER JOIN products _
                   ON categories.categoryid = products.categoryid _
                   ORDER BY products.categoryid, products.productid"
        m_cnn = New OleDb.OleDbConnection(m_cnnString)
        Dim m_Cmd As New OleDb.OleDbCommand(sqlString, m_cnn)
        If m_cnn.State = ConnectionState.Closed Then
            m_cnn.Open()

         End If
        m_reader = m_Cmd.ExecuteReader()

End Sub

[C#]

private static OleDbConnection m_cnn;
private static OleDbDataReader  m_reader;
private void rptUnbound_ReportStart(object sender, System.EventArgs
         eArgs)
{
         string m_dbPath = getDatabasePath();
         string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
                  Source=C:\Program Files\Data Dynamics\ActiveReports.NET\
                  Data\NWIND.MDB;Persist Security Info=False";
         string sqlString = "SELECT * FROM categories INNER JOIN
                  products ON categories.categoryid
                  = products.categoryid ORDER BY products.categoryid,
                  products.productid";
         m_cnn = new OleDbConnection(m_cnnString);
         OleDbCommand m_Cmd = new OleDbCommand(sqlString,m_cnn);
         if(m_cnn.State == ConnectionState.Closed)
         {
                  m_cnn.Open();
         }
         m_reader = m_Cmd.ExecuteReader();
}
```

Adding controls to the report to contain data

To add controls to the report

- Add the following controls to the Detail section of rptUnbound:

Control	DataField	Name	Text/Caption	Location
TextBox	ProductName	**txtProductName**	Product Name	0.0625, 0.375
TextBox	UnitsInStock	**txtUnitsInStock**	Units In Stock	4.75, 0.375
Label	(Empty string)	**lblProductName**	Product Name	0.0625, 0.0625
TextBox	Description	**txtDescription**	Description	1.8125, 0.6875
Label	(Empty string)	**lblCategoryDescription**	Category Description	0.0625. 0.6875
Label	(Empty string)	**lblUnitsInStock**	Units In Stock	4.75, 0.0625
Label	(Empty string)	**lblCategoryName**	Category Name	3.125, 0.0625
TextBox	CategoryName	**txtCategoryName**	Category Name	3.125, 0.375

Line	(Empty string)	**Line1**	(Empty string)	X1 = 0
				Y1 = 1.0625
				X2 = 6.5
				Y2 = 1.0625

Using the DataInitialize event to add fields

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnbound, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnbound, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *DataInitialize*. This creates an event-handling method for rptUnbound's DataInitialize event. Add code to the handler to:

- Add fields to the report's fields collection

To write the code in C#

- Click in the gray area below rptUnbound to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *DataInitialize*. This creates an event-handling method for rptUnbound's DataInitialize event. Add code to the handler to:

    - Add fields to the report's fields collection

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptUnbound_DataInitialize(ByVal sender As Object, _
         ByVal e As System.EventArgs) Handles MyBase.DataInitialize
        Fields.Add("CategoryName")
        Fields.Add("ProductName")
        Fields.Add("UnitsInStock")
        Fields.Add("Description")
End Sub

[C#]

private void rptUnbound_DataInitialize(object sender, System.EventArgs
         eArgs)
{
         Fields.Add("CategoryName");
         Fields.Add("ProductName");
         Fields.Add("UnitsInStock");
         Fields.Add("Description");
}
```

Using the FetchData event to populate the report fields

To write the code in Visual Basic

- Right-click in any section of the design window of rptUnbound, and click on **View Code** to display the code view for the report. At the top left of the code view for rptUnbound, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *FetchData*. This creates an event-handling method for rptUnbound's FetchData event. Add code to the handler to:

    - Retrieve information to populate the report fields

To write the code in C#

- Click in the gray area below rptUnbound to select the report. Click on the events icon in the **Properties** window to display available events for the report. Double-click *FetchData*. This creates an event-handling method for rptUnbound's FetchData event. Add code to the handler to:

    - Retrieve information to populate the report fields

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub rptUnbound_FetchData(ByVal sender As Object, _
         ByVal eArgs As DataDynamics.ActiveReports.ActiveReport._
         FetchEventArgs) Handles MyBase.FetchData
        Try
            m_reader.Read()
            Me.Fields("CategoryName").Value = m_reader("CategoryName")
            Me.Fields("ProductName").Value = m_reader("ProductName")
            Me.Fields("UnitsInStock").Value = m_reader("UnitsInStock")
            Me.Fields("Description").Value = m_reader("Description")
        eArgs.EOF = False
        Catch ex As Exception
             eArgs.EOF = True
        End Try
    End Sub

[C#]

private void rptUnbound_FetchData(object sender,
         DataDynamics.ActiveReports.ActiveReport.
         FetchEventArgs eArgs)
{
         try
         {
                  m_reader.Read();
                  Fields["CategoryName"].Value = m_reader
                           ["CategoryName"].ToString();
                  Fields["ProductName"].Value = m_reader
                           ["ProductName"].ToString();
                  Fields["UnitsInStock"].Value = m_reader
                           ["UnitsInStock"].ToString();
                  Fields["Description"].Value = m_reader
                           ["Description"].ToString();
                  eArgs.EOF = false;
         }
         catch
         {
                  eArgs.EOF = true;
         }
}
```

Professional Edition Walkthroughs

ActiveReports for .NET Professional Edition allows developers to create and deploy custom report designers and controls on the web. The following walkthroughs demonstrate how effectively to use specific features of the Professional Edition.

In this section

- Creating an End-User Report Designer
- Deploying ActiveReports Web Applications
- HTTP Handlers
- Web Viewer Control

Creating an End-User Report Designer Walkthroughs

ActiveReports allows you to host the ActiveReports End-User Report Designer control in your application and provide end-user report editing capabilities. The control's methods and properties provide easy access to save and load report layouts, monitor and control the design environment and customize the look and feel to satisfy the needs of your end users.

- Creating the Basic Layout for an End-User Report Designer
- Configuring the ActiveReports Toolbox
- Configuring the Layout Toolbar
- Configuring the Report Toolbar
- Adding a Viewer Control for the End-User Report Designer

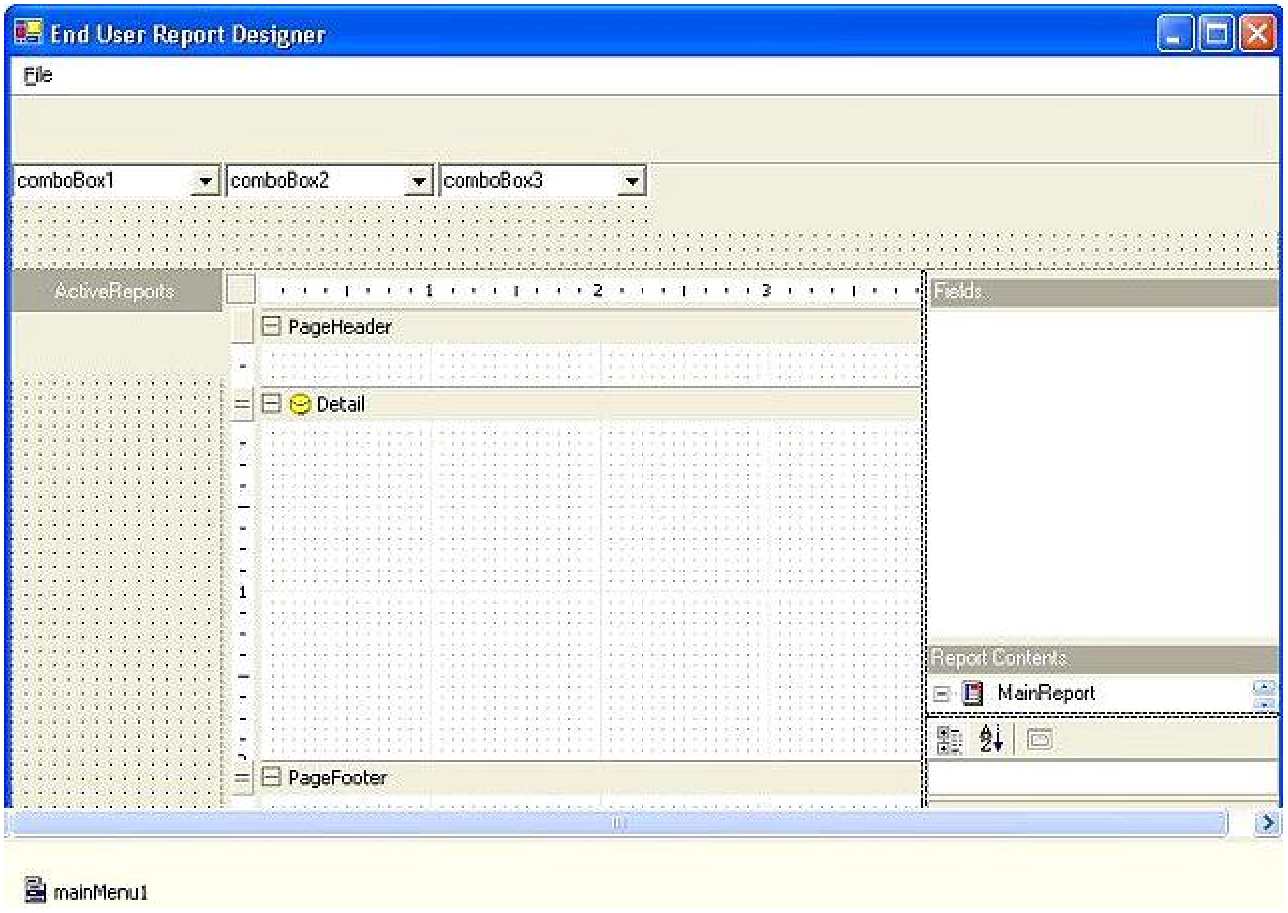
Creating the Basic Layout for an End-User Report Designer

This walkthrough illustrates how to set up a basic layout of an End-User Report Designer on a Windows Form.

This walkthrough is split up into the following activities:

- Adding a Windows Form to a Visual Studio project
- Adding panels and splitters to the form
- Adding the ActiveReports End-User Report Designer control to the form
- Adding the Report Explorer to the form
- Adding the Property Grid to the form
- Adding a label to the form
- Adding toolbars and combo boxes to the form
- Adding a Main Menu control to the form

When you have completed this walkthrough, you will have a basic layout which looks similar to the following.



Adding a Windows Form to a Visual Studio project

To add a Windows Form to a Visual Studio project

1. Open a Visual Studio project.
2. Click on **Project > Add Windows Form...**
3. Name the form frmDesigner and click **Open**.
4. Resize frmDesigner to the desired height and width.
5. Change the text for frmDesigner to "End-User Report Designer."

Adding panels and splitters to the form

To add panels and splitters to the form

1. Add a panel to the form and set the Dock property to Top.
2. Add a second panel to the form and set the Dock property to Left.
3. Add a third panel to the form and set the Dock property to Right.
4. Add a splitter to the form and set the Dock property to Right.
5. In the right panel, add a panel and set the Dock property to Bottom.
6. Add a splitter to the right panel and set the Dock property to Bottom.
7. Add a panel to the right panel and set the Dock property to Fill.

Adding the ActiveReports End-User Report Designer control to the form

To add the control

1. Click at the top of frmDesigner to select the form.
2. Click on the ActiveReports Designer control from the toolbox and drag it onto the form.
3. Set the Dock property to Fill.

Adding the Report Explorer to the form

To add the Report Explorer

1. Click on the top right panel to select it.
2. Click on the ActiveReports ReportExplorer control from the toolbox and drag it onto the panel.
3. Set the Dock property to Fill.
4. Set the Report Designer property to "designer1."

Adding the Property Grid to the form

To add the Property Grid

1. Right-click on the toolbox where you wish to add the Property Grid control.
2. Click on **Customize Toolbox...**
3. Select the .NET Framework Components tab.
4. Select PropertyGrid from the list and click OK.
5. Click on the bottom right panel to select it.
6. Click on the Property Grid icon and drag it onto the panel.
7. Set the Dock property to Fill.

Adding a label to the form

To add a label

1. Add a label to the left panel.
2. Set the Dock property to Top.
3. Change the BackColor to Control Dark.
4. Change the ForeColor to Control Light Light.
5. Change the Text to "ActiveReports."

Adding Toolbars and Combo Boxes to the form

To add toolbars and combo boxes

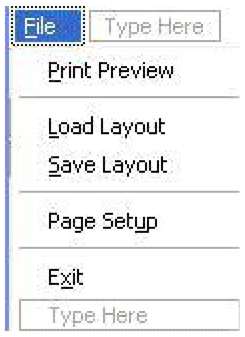
1. Click on the top panel to select it and add a toolbar to the panel.
2. Change the name of the toolbar to tlbLayout.
3. Click on the left panel to select it and add a toolbar to the panel.
4. Set the Dock property to Fill.
5. Change the name of the toolbar to tlbARToolbox.

6. Add three combo boxes to the top panel underneath tlbLayout.
7. Set the Dock property to Left for all three combo boxes and change the names to: cmbClassName, cmbFonts and cmbFontSize.
8. Add a toolbar to the top panel underneath tlbLayout.
9. Set the Dock property to Fill.
10. Change the name of the toolbar to tlbReport.

Adding a Main Menu control to the form

To add a Main Menu control

1. Click on frmDesigner to select it.
2. Click on the MainMenu control and drag it onto the form.
3. Set the Menu property for frmDesigner to MainMenu1.
4. Add the following items to the menu:
   - &File--change the name to mnuFile
   - &Print Preview--change the name to mnuPrintPreview
   - -
   - &Load Layout--change the name to mnuLoadLayout
   - &Save Layout--change the name to mnuSaveLayout
   - -
   - Page Set&up--change the name to mnuPageSetup
   - -
   - E&xit--change the name to mnuExit



Adding Code for the End-User Report Designer

This walkthrough is split up into the following activities:

- Adding a protected enumeration to frmDesigner
- Adding code to the frmDesigner_Load event
- Adding code to the Designer1_SelectionChanged event

Adding a protected enumeration to frmDesigner

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code just below the "Windows Form Designer generated code" region.

To write the code in C#

- Double-click anywhere on frmDesigner to show the code view for the Windows Form. Add the following code just below the "Windows Form Designer generated code" region.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Protected Enum toolbarModes
        singleControl
        twoControls
        multiControls
        noControls
End Enum

[C#]

protected enum toolbarModes
{
         singleControl,
         twoControls,
         multiControls,
         noControls,
}
```

Adding code to the frmDesigner_Load event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the frmDesigner_Load event.

To write the code in C#

- Click at the top of frmDesigner to select the Windows Form. Click on the events icon in the **Properties** window to display available events for the section. Double-click *Load*. This creates an event-handling method for the frmDesigner_Load event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Dim i As Integer
Dim cnt As Integer
Dim ctl As String
Private Sub frmDesigner_Load(ByVal sender As Object, ByVal _
         e As System.EventArgs) Handles MyBase.Load
        Me.fillClassName()
        Me.fillFonts()
        Me.fillFontSizes()
        Me.setModes(toolbarModes.noControls)
        Me.Designer1.Focus()
End Sub

[C#]

private void frmDesigner_Load(object sender, System.EventArgs e)
{
```

```
         this.fillClassName();
         this.fillFonts();
         this.fillFontSizes();
         this.setModes(toolbarModes.noControls);
         this.designer1.Focus();
}
```

Adding code to the Designer1_SelectionChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *designer1*. At the top right of the code window, click the drop-down arrow and select *SelectionChanged*. This creates an event-handling method for the Designer1_SelectionChanged event.

To write the code in C#

- Click in the designer window of frmDesigner to select Designer1. Click on the events icon in the **Properties** window to display available events for the section. Double-click *SelectionChanged*. This creates an event-handling method for the Designer1_SelectionChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Designer1_SelectionChanged() Handles _
         Designer1.SelectionChanged
        selChangeReportToolbar()
        selChangeLayoutToolbar()
        selChangePropGrid()
End Sub

[C#]

private void designer1_SelectionChanged()
{
         selChangeReportToolbar();
         selChangeLayoutToolbar();
         selChangePropGrid();
}
```

Adding Code for the Main Menu

This walkthrough is split up into the following activities:

- Adding code for the Print Preview menu item
- Adding code for the Load Layout menu item
- Adding code for the Save Layout menu item
- Adding code for the Page Setup menu item
- Adding code for the Exit menu item

Adding code for the Print Preview menu item

To write the code in Visual Basic

- On the Main Menu, double-click the entry for Print Preview. This creates an event-handling method for mnuPrintPreview's Click event. Add the following code to the mnuPrintPreview_Click event.

To write the code in C#

- On the Main Menu, double-click the entry for Print Preview. This creates an event-handling method for mnuPrintPreview's Click event. Add the following code to the mnuPrintPreview_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub mnuPrintPreview_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles mnuPrintPreview.Click
        Dim rpt As New DataDynamics.ActiveReports.ActiveReport()
        Dim m_stream As New System.IO.MemoryStream()
        Me.Designer1.Report.SaveLayout(m_stream)
        m_stream.Position = 0
        rpt.LoadLayout(m_stream)
        m_stream.Close()
        Dim frm As New frmViewer()
        frm.SetReport(rpt)
        frm.ShowDialog(Me)
End Sub

[C#]

private void mnuPrintPreview_Click(object sender, System.EventArgs e)
{
         DataDynamics.ActiveReports.ActiveReport rpt = new
                  DataDynamics.ActiveReports.ActiveReport();
         System.IO.MemoryStream m_stream = new System.IO.MemoryStream();
         this.designer1.Report.SaveLayout(m_stream);
         m_stream.Position = 0;
         rpt.LoadLayout(m_stream);
         m_stream.Close();
         frmViewer frm = new frmViewer();
         frm.SetReport(rpt);
         frm.ShowDialog(this);
}
```

Adding code for the Load Layout menu item

To write the code in Visual Basic

- On the Main Menu, double-click the entry for Load Layout. This creates an event-handling method for mnuLoadLayout's Click event. Add the following code to the mnuLoadLayout_Click event.

To write the code in C#

- On the Main Menu, double-click the entry for Load Layout. This creates an event-handling method for mnuLoadLayout's Click event. Add the following code to the mnuLoadLayout_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub mnuLoadLayout_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles mnuLoadLayout.Click
        Me.Designer1.ExecuteAction(DesignerAction.FileOpen)
End Sub

[C#]

private void mnuLoadLayout_Click(object sender, System.EventArgs e)
{
         this.designer1.ExecuteAction(DataDynamics.ActiveReports.
                  Design.DesignerAction.FileOpen);
}
```

Adding code for the Save Layout menu item

To write the code in Visual Basic

- On the Main Menu, double-click the entry for Save Layout. This creates an event-handling method for mnuSaveLayout's Click event. Add the following code to the mnuSaveLayout_Click event.

To write the code in C#

- On the Main Menu, double-click the entry for Save Layout. This creates an event-handling method for mnuSaveLayout's Click event. Add the following code to the mnuSaveLayout_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub mnuSaveLayout_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles mnuSaveLayout.Click
        Me.Designer1.ExecuteAction(DesignerAction.FileSave)
End Sub

[C#]

private void mnuSaveLayout_Click(object sender, System.EventArgs e)
{
         this.designer1.ExecuteAction(DataDynamics.ActiveReports.
                  Design.DesignerAction.FileSave);
}
```

Adding code for the Page Setup menu item

To write the code in Visual Basic

- On the Main Menu, double-click the entry for Page Setup. This creates an event-handling method for mnuPageSetup's Click event. Add the following code to the mnuPageSetup_Click event.

To write the code in C#

- On the Main Menu, double-click the entry for Page Setup. This creates an event-handling method for mnuPageSetup's Click event. Add the following code to the mnuPageSetup_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub mnuPageSetup_Click(ByVal sender As System.Object, _
         ByVal e As System.EventArgs) Handles mnuPageSetup.Click
        Me.Designer1.ExecuteAction(DesignerAction.ReportSettings)
End Sub

[C#]

private void mnuPageSetup_Click(object sender, System.EventArgs e)
{
         this.designer1.ExecuteAction(DataDynamics.ActiveReports.
                   Design.DesignerAction.ReportSettings);
}
```

Adding code for the Exit menu item

To write the code in Visual Basic

- On the Main Menu, double-click the entry for Exit. This creates an event-handling method for mnuExit's Click event. Add the following code to the mnuExit_Click event.

To write the code in C#

- On the Main Menu, double-click the entry for Exit. This creates an event-handling method for mnuExit's Click event. Add the following code to the mnuExit_Click event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub mnuExit_Click(ByVal sender As System.Object, ByVal _
         e As System.EventArgs) Handles mnuExit.Click
        Me.Close()
End Sub

[C#]

private void mnuExit_Click(object sender, System.EventArgs e)
{
         this.Close();
}
```

Adding Code for the Property Grid

This walkthrough is made up of the following activity:

- Adding code for the SelectionChanged event for the property grid

Adding code to the selChangePropGrid event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the selChangePropGrid event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the selChangePropGrid event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub selChangePropGrid()
        cnt = Me.Designer1.Selection.Count - 1
        Dim selectedControls(cnt) As Object
        Try
            For i = 0 To Me.Designer1.Selection.Count - 1
                selectedControls.SetValue(CType(Me.Designer1._
                            Selection(i), Object), i)
            Next
            Me.PropertyGrid1.SelectedObjects = selectedControls
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message + ": " + ex.Source + ": _
                   " + ex.StackTrace, "Selection Error", _
                   MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
End Sub

[C#]

private void selChangePropGrid()
{
int cnt = this.designer1.Selection.Count;
         System.Object[] selectedControls = new object[cnt];
         try
         {
                  for(int i=0;i<this.designer1.Selection.Count;i++)
                  {
                           selectedControls.SetValue((System.Object)this.designer1.
                  Selection[i],i);
                  }
                  this.propertyGrid1.SelectedObjects = selectedControls;
         }
         catch(Exception ex)
         {
                  MessageBox.Show(this,ex.Message + ": " + ex.Source
                           + ": " + ex.StackTrace, "Selection
                           Error",MessageBoxButtons.OK,
                           MessageBoxIcon.Error);
         }
}
```

Configuring the ActiveReports Toolbox

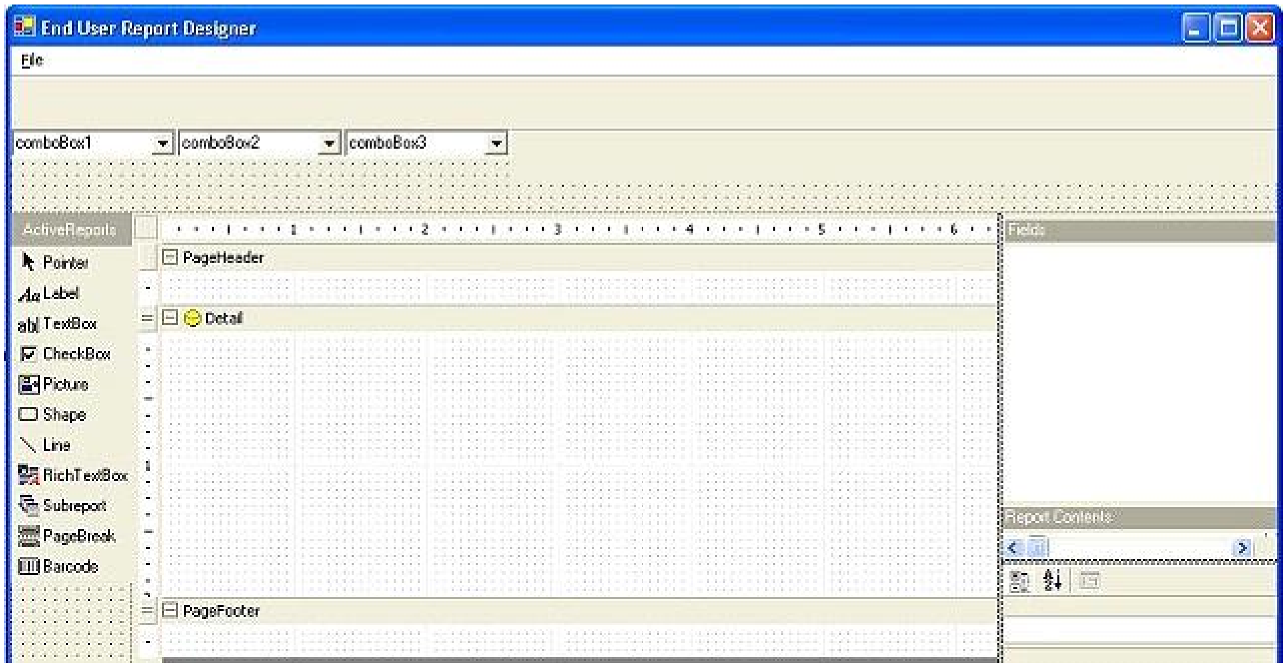
This walkthrough illustrates how to set up the ActiveReports Toolbox to add to the basic layout of your End-User Report Designer. This walkthrough builds on the walkthrough "Creating the Basic Layout for an End-User Report Designer."

This walkthrough is split up into the following activities:

- Adding buttons to the Toolbox collection
- Adding an ImageList to add icons for the Toolbox

- Adding the icon images to the ImageList
- Setting properties for the buttons in the Toolbox collection

When you have completed this walkthrough, your layout for the End-User Report Designer will look similar to the following.



Adding buttons to the Toolbox collection

To add buttons to the Toolbox collection

1. Click on tlbARToolbox to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. Add eleven buttons to the collection and click **OK**.

Adding an ImageList to add icons

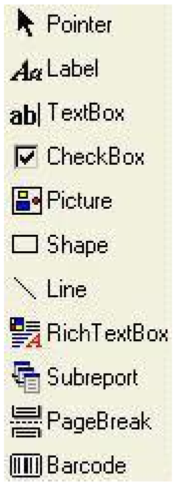
To add an ImageList

1. Click at the top of frmDesigner to select it.
2. Click on the ImageList control in the toolbox and drag it onto the form.
3. Change the name of ImageList to ImgToolbox.
4. Set the ImageList property for tlbARToolbox to ImgToolbox.

Adding the icon images to the ImageList

To add the icon images

1. Click on ImgToolbox at the bottom of frmDesigner to select it.
2. Click on the ellipsis for the Images (collection) property.
3. Add the following icons to the ImgToolbox:



4. Click **OK** to continue.

Setting properties for the buttons in the Toolbox collection

To set properties for the buttons

1. Click on tlbARToolbox to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. For each button in the collection, make the following property changes and add the appropriate icons to the ImageIndex.

Button #	Tag	Name	Text	ToolTipText
toolBarButton1	Pointer	**tbbPointer**	Pointer	Pointer
toolBarButton2	Label	**tbbLabel**	Label	Label
toolBarButton3	TextBox	**tbbTextBox**	TextBox	TextBox
toolBarButton4	CheckBox	**tbbCheckBox**	CheckBox	CheckBox
toolBarButton5	Picture	**tbbPicture**	Picture	Picture
toolBarButton6	Shape	**tbbShape**	Shape	Shape
toolBarButton7	Line	**tbbLine**	Line	Line
toolBarButton8	RichText	**tbbRichTextBox**	RichTextBox	RichTextBox
toolBarButton9	Subreport	**tbbSubreport**	Subreport	Subreport
toolBarButton10	PageBreak	**tbbPageBreak**	PageBreak	PageBreak
toolBarButton11	Barcode	**tbbBarcode**	Barcode	Barcode

4. Click **OK** to continue.
5. Set the Appearance property for tlbARToolbox to Flat.
6. Change the TextAlign property for tlbARToolbox to Right.
7. Resize Panel2 to fit one button on each line.

Adding Code for the ActiveReports Toolbox

This walkthrough is made up of the following activity:

- Adding code for the ActiveReports Toolbox ButtonClick event

Adding code for the tlbARToolbox ButtonClick event

To write the code in Visual Basic

- Double-click on tlbARToolbox. This creates an event-handling method for tlbARToolbox's ButtonClick event. Add the following code to the tlbARToolbox_ButtonClick event.

To write the code in C#

- Double-click on tlbARToolbox. This creates an event-handling method for tlbARToolbox's ButtonClick event. Add the following code to the tlbARToolbox_ButtonClick event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub tlbARToolbox_ButtonClick(ByVal sender As _
         System.Object, ByVal e As System.Windows.Forms._
         ToolBarButtonClickEventArgs) Handles tlbARToolbox.ButtonClick
        If (e.Button.Tag.ToString() <> "Pointer") Then
            Designer1.ToolBoxItem = e.Button.Tag.ToString()
        Else
            Designer1.ToolBoxItem = Nothing
        End If
End Sub

[C#]

private void tlbARToolbox_ButtonClick(object sender,
         System.Windows.Forms.
         ToolBarButtonClickEventArgs e)
{
         if(e.Button.Tag.ToString() != "Pointer")
         {
                  designer1.ToolBoxItem = e.Button.Tag.ToString();
         }
         else
         {
                  designer1.ToolBoxItem = null;
         }
}
```

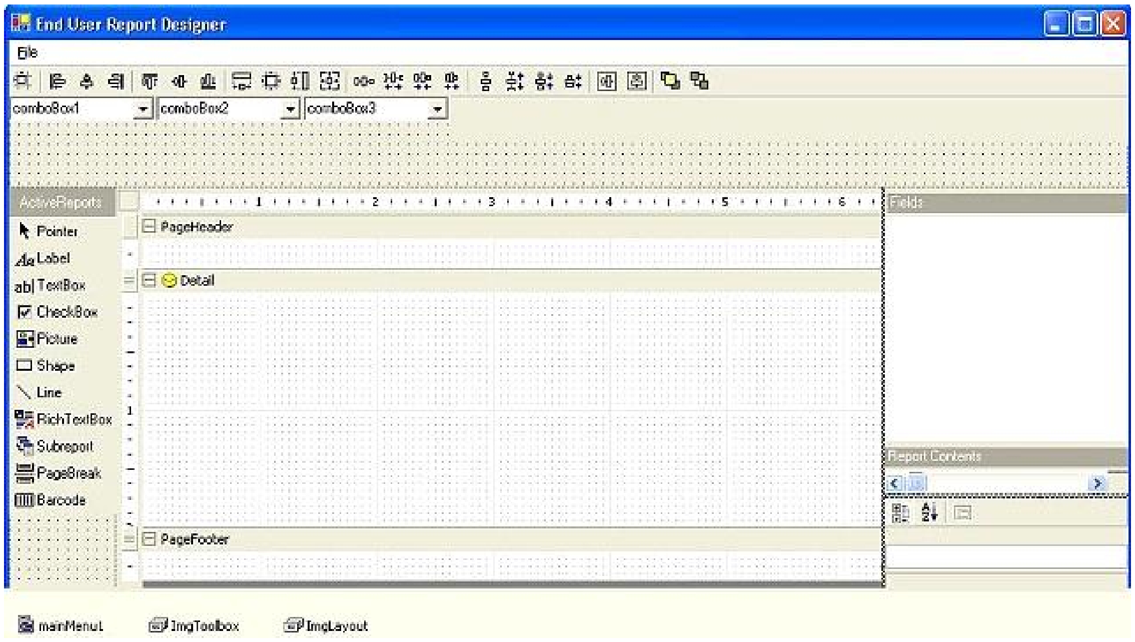
Configuring the Layout Toolbar

This walkthrough illustrates how to set up the ActiveReports Layout Toolbar to add to the basic layout of your End-User Report Designer. This walkthrough builds on the walkthroughs "Creating the Basic Layout for an End-User Report Designer" and "Configuring the ActiveReports Toolbox."

This walkthrough is split up into the following activities:

- Adding buttons to the Toolbar collection
- Adding an ImageList to add icons for the Toolbar
- Adding the icon images to the ImageList
- Setting properties for the buttons in the Toolbar collection

When you have completed this walkthrough your layout for End-User Report Designer will look similar to the following.



Adding buttons to the Toolbar collection

To add buttons to the Toolbar collection

1. Click on tlbLayout to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. Add thirty buttons to the collection and click **OK**.

Adding an ImageList to add icons

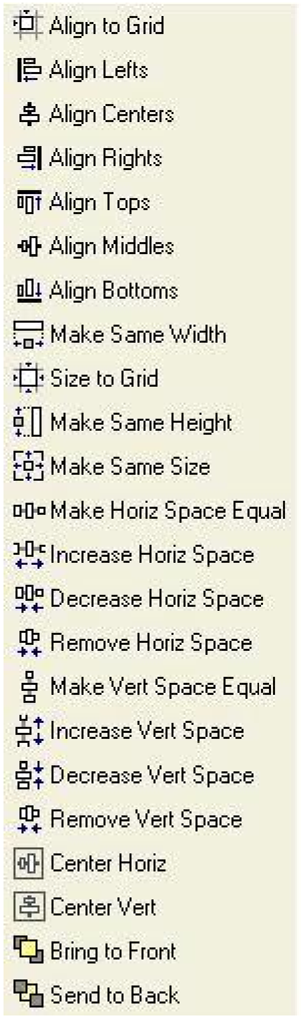
To add an ImageList

1. Click at the top of frmDesigner to select it.
2. Click on the ImageList control in the toolbox and drag it onto the form.
3. Change the name of ImageList to ImgLayout.
4. Set the ImageList property for tlbLayout to ImgLayout.

Adding the icon images to the ImageList

To add the icon images

1. Click on ImgLayout at the bottom of frmDesigner to select it.
2. Click on the ellipsis for the Images (collection) property.
3. Add the following icons to the ImgLayout:



4. Click **OK** to continue.

Setting properties for the buttons in the Toolbox collection

To set properties for the buttons

1. Click on tlbLayout to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. For each button in the collection, make the following property changes and add the appropriate icons to the ImageIndex.

#	Tag	Name	Style	ToolTipText
1	AligntoGrid	**tbbAligntoGrid**	PushButton	AligntoGrid
2	(Empty string)	**tbbLine1**	Separator	(Empty string)
3	AlignLefts	**tbbAlignLefts**	PushButton	AlignLefts
4	AlignCenters	**tbbAlignCenters**	PushButton	AlignCenters
5	AlignRights	**tbbAlignRights**	PushButton	AlignRights
6	(Empty string)	**tbbLine2**	Separator	(Empty string)

7	AlignTops	**tbbAlignTops**	PushButton	AlignTops
8	AlignMiddles	**tbbAlignMiddles**	PushButton	AlignMiddles
9	AlignBottoms	**tbbAlignBottoms**	PushButton	AlignBottoms
10	(Empty string)	**tbbLine3**	Separator	(Empty string)
11	MakeSameWidth	**tbbMakeSameWidth**	PushButton	MakeSameWidth
12	SizeToGrid	**tbbSizeToGrid**	PushButton	SizeToGrid
13	MakeSameHeight	**tbbMakeSameHeight**	PushButton	MakeSameHeight
14	MakeSameSize	**tbbMakeSameSize**	PushButton	MakeSameSize
15	(Empty string)	**tbbLine4**	Separator	(Empty string)
16	MakeHorizSpaceEqual	**tbbMakeHorizSpaceEqual**	PushButton	MakeHorizSpaceEqual
17	IncreaseHorizSpace	**tbbIncreaseHorizSpace**	PushButton	IncreaseHorizSpace
18	DecreaseHorizSpace	**tbbDecreaseHorizSpace**	PushButton	DecreaseHorizSpace
19	RemoveHorizSpace	**tbbRemoveHorizSpace**	PushButton	RemoveHorizSpace
20	(Empty string)	**tbbLine5**	Separator	(Empty string)
21	MakeVertSpaceEqual	**tbbMakeVertSpaceEqual**	PushButton	MakeVertSpaceEqual
22	IncreaseVertSpace	**tbbIncreaseVertSpace**	PushButton	IncreaseVertSpace
23	DecreaseVertSpace	**tbbDecreaseVertSpace**	PushButton	DecreaseVertSpace
24	RemoveVertSpace	**tbbRemoveVertSpace**	PushButton	RemoveVertSpace
25	(Empty string)	**tbbLine6**	Separator	(Empty string)
26	CenterHoriz	**tbbCenterHoriz**	PushButton	CenterHoriz
27	CenterVert	**tbbCenterVert**	PushButton	CenterVert
28	(Empty string)	**tbbLine7**	Separator	(Empty string)
29	BringtoFront	**tbbBringtoFront**	PushButton	BringtoFront
30	SendtoBack	**tbbSendtoBack**	PushButton	SendtoBack

4. Click **OK** to continue.
5. Set the Appearance property for tlbLayout to Flat.

Adding Code for the Layout Toolbar

This walkthrough is split up into the following activities:

- Adding code for the Layout toolbar's SelectionChanged event
- Adding code to set the modes for the Layout toolbar
- Adding code for the Layout toolbar's executeLayoutAction event
- Adding code for the Layout toolbar's ButtonClick event

Adding code for the Layout toolbar's SelectionChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the selChangeLayoutToolbar event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the selChangeLayoutToolbar event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub selChangeLayoutToolbar()
        cnt = Me.Designer1.Selection.Count
        If cnt = 0 Then
            Me.setModes(toolbarModes.noControls)
        ElseIf ((cnt = 1) AndAlso ((Me.Designer1._
                   Selection(0).GetType().BaseType.ToString() _
                   <> "DataDynamics.ActiveReports.Section") AndAlso_
                   (Me.Designer1.Selection(0).GetType().BaseType._
                   ToString() <> "System.Object"))) Then
            Me.setModes(toolbarModes.singleControl)
        ElseIf (cnt = 2) Then
            Me.setModes(toolbarModes.twoControls)
        ElseIf (cnt > 2) Then
            Me.setModes(toolbarModes.multiControls)
        End If
End Sub

[C#]

private void selChangeLayoutToolbar()
{
         int cnt = this.designer1.Selection.Count;
         if(cnt==0)
         {
                  this.setModes(toolbarModes.noControls);
         }
         else if((cnt==1)&&((this.designer1.Selection[0].
                  GetType().BaseType.ToString() !=
                  "DataDynamics.ActiveReports.
                  Section")&&(this.designer1.Selection[0].GetType().
                  BaseType.ToString() != "System.Object")))
         {
                  this.setModes(toolbarModes.singleControl);
         }
         else if(cnt==2)
         {
                  this.setModes(toolbarModes.twoControls);
         }
         else if(cnt >2)
         {
                  this.setModes(toolbarModes.multiControls);
         }
}
```

Adding code to set the modes for the Layout toolbar

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the setModes event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the setModes event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Private Sub setModes(ByVal tbModes As Form1.toolbarModes)
        Me.SuspendLayout()
        Select Case tbModes
            Case toolbarModes.multiControls
                Me.tbbAlignBottoms.Enabled = True
                Me.tbbAlignCenters.Enabled = True
                Me.tbbAlignLefts.Enabled = True
                Me.tbbAlignMiddles.Enabled = True
                Me.tbbAlignRights.Enabled = True
                Me.tbbAligntoGrid.Enabled = True
                Me.tbbAlignTops.Enabled = True
                Me.tbbBringToFront.Enabled = True
                Me.tbbCenterHoriz.Enabled = True
                Me.tbbCenterVert.Enabled = True
                Me.tbbDecreaseHorizSpace.Enabled = True
                Me.tbbDecreaseVertSpace.Enabled = True
                Me.tbbIncreaseHorizSpace.Enabled = True
                Me.tbbIncreaseVertSpace.Enabled = True
                Me.tbbMakeHorizSpaceEqual.Enabled = True
                Me.tbbMakeSameHeight.Enabled = True
                Me.tbbMakeSameSize.Enabled = True
                Me.tbbMakeSameWidth.Enabled = True
                Me.tbbMakeVertSpaceEqual.Enabled = True
                Me.tbbRemoveHorizSpace.Enabled = True
                Me.tbbRemoveVertSpace.Enabled = True
                Me.tbbSendToBack.Enabled = True
                Me.tbbSizeToGrid.Enabled = True
            Case toolbarModes.twoControls
                Me.tbbAlignBottoms.Enabled = True
                Me.tbbAlignCenters.Enabled = True
                Me.tbbAlignLefts.Enabled = True
                Me.tbbAlignMiddles.Enabled = True
                Me.tbbAlignRights.Enabled = True
                Me.tbbAligntoGrid.Enabled = True
                Me.tbbAlignTops.Enabled = True
                Me.tbbBringToFront.Enabled = True
                Me.tbbCenterHoriz.Enabled = True
                Me.tbbCenterVert.Enabled = True
                Me.tbbDecreaseHorizSpace.Enabled = False
                Me.tbbDecreaseVertSpace.Enabled = False
                Me.tbbIncreaseHorizSpace.Enabled = False
                Me.tbbIncreaseVertSpace.Enabled = False
                Me.tbbMakeHorizSpaceEqual.Enabled = False
                Me.tbbMakeSameHeight.Enabled = True
                Me.tbbMakeSameWidth.Enabled = True
                Me.tbbMakeSameSize.Enabled = True
                Me.tbbMakeVertSpaceEqual.Enabled = False
                Me.tbbRemoveHorizSpace.Enabled = True
                Me.tbbRemoveVertSpace.Enabled = True
                Me.tbbSendToBack.Enabled = True
                Me.tbbSizeToGrid.Enabled = True
            Case toolbarModes.singleControl
                Me.tbbAlignBottoms.Enabled = False
                Me.tbbAlignCenters.Enabled = False
                Me.tbbAlignLefts.Enabled = False
                Me.tbbAlignMiddles.Enabled = False
                Me.tbbAlignRights.Enabled = False
                Me.tbbAligntoGrid.Enabled = True
                Me.tbbAlignTops.Enabled = False
                Me.tbbCenterHoriz.Enabled = True
                Me.tbbCenterVert.Enabled = True
                Me.tbbDecreaseVertSpace.Enabled = False
                Me.tbbIncreaseVertSpace.Enabled = False
                Me.tbbDecreaseHorizSpace.Enabled = False
                Me.tbbIncreaseHorizSpace.Enabled = False
                Me.tbbMakeHorizSpaceEqual.Enabled = False
                Me.tbbMakeSameHeight.Enabled = False
                Me.tbbMakeSameSize.Enabled = False
                Me.tbbMakeSameWidth.Enabled = False
                Me.tbbMakeVertSpaceEqual.Enabled = False
                Me.tbbRemoveHorizSpace.Enabled = False
                Me.tbbRemoveVertSpace.Enabled = False
                Me.tbbSendToBack.Enabled = True
                Me.tbbSizeToGrid.Enabled = True
                Me.tbbBringToFront.Enabled = True
            Case toolbarModes.noControls
                Me.tbbAlignBottoms.Enabled = False
                Me.tbbAlignCenters.Enabled = False
                Me.tbbAlignLefts.Enabled = False
                Me.tbbAlignMiddles.Enabled = False
                Me.tbbAlignRights.Enabled = False
                Me.tbbAligntoGrid.Enabled = False
                Me.tbbAlignTops.Enabled = False
                Me.tbbBringToFront.Enabled = False
                Me.tbbCenterHoriz.Enabled = False
                Me.tbbCenterVert.Enabled = False
                Me.tbbDecreaseHorizSpace.Enabled = False
                Me.tbbDecreaseVertSpace.Enabled = False
                Me.tbbIncreaseHorizSpace.Enabled = False
                Me.tbbIncreaseVertSpace.Enabled = False
                Me.tbbMakeHorizSpaceEqual.Enabled = False
                Me.tbbMakeSameHeight.Enabled = False
                Me.tbbMakeSameSize.Enabled = False
                Me.tbbMakeSameWidth.Enabled = False
                Me.tbbMakeVertSpaceEqual.Enabled = False
                Me.tbbRemoveHorizSpace.Enabled = False
                Me.tbbRemoveVertSpace.Enabled = False
                Me.tbbSendToBack.Enabled = False
                Me.tbbSizeToGrid.Enabled = False
        End Select
        Me.ResumeLayout()
End Sub

[C#]

private void setModes(frmDesigner.toolbarModes tbModes)
{
         this.SuspendLayout();
         switch(tbModes)
         {
                  case toolbarModes.multiControls:
                           this.tbbAlignBottoms.Enabled =true;
                           this.tbbAlignCenters.Enabled = true;
                           this.tbbAlignLefts.Enabled = true;
                           this.tbbAlignMiddles.Enabled = true;
                           this.tbbAlignRights.Enabled = true;
                           this.tbbAlignToGrid.Enabled = true;
                           this.tbbAlignTops.Enabled = true;
                           this.tbbBringToFront.Enabled = true;
                           this.tbbCenterHoriz.Enabled = true;
                           this.tbbCenterVert.Enabled = true;
                           this.tbbDecreaseHorizSpace.Enabled = true;
                           this.tbbDecreaseVertSpace.Enabled = true;
                           this.tbbIncreaseHorizSpace.Enabled = true;
                           this.tbbIncreaseVertSpace.Enabled = true;
                           this.tbbMakeHorizSpaceEqual.Enabled = true;
                           this.tbbMakeSameHeight.Enabled = true;
                           this.tbbMakeSameSize.Enabled = true;
                           this.tbbMakeSameWidth.Enabled = true;
                           this.tbbMakeVertSpaceEqual.Enabled = true;
                           this.tbbRemoveHorizSpace.Enabled = true;
                           this.tbbRemoveVertSpace.Enabled = true;
                           this.tbbSendToBack.Enabled = true;
                           this.tbbSizeToGrid.Enabled = true;
                           break;
                  case toolbarModes.twoControls:
                           this.tbbAlignBottoms.Enabled =true;
                           this.tbbAlignCenters.Enabled = true;
                           this.tbbAlignLefts.Enabled = true;
                           this.tbbAlignMiddles.Enabled = true;
                           this.tbbAlignRights.Enabled = true;
                           this.tbbAlignToGrid.Enabled = true;
                           this.tbbAlignTops.Enabled = true;
                           this.tbbBringToFront.Enabled = true;
                           this.tbbCenterHoriz.Enabled = true;
                           this.tbbCenterVert.Enabled = true;
                           this.tbbDecreaseHorizSpace.Enabled = false;
                           this.tbbDecreaseVertSpace.Enabled = false;
                           this.tbbIncreaseHorizSpace.Enabled = false;
                           this.tbbIncreaseVertSpace.Enabled = false;
                           this.tbbMakeHorizSpaceEqual.Enabled = false;
                           this.tbbMakeSameHeight.Enabled = true;
                           this.tbbMakeSameSize.Enabled = true;
                           this.tbbMakeSameWidth.Enabled = true;
                           this.tbbMakeVertSpaceEqual.Enabled = false;
                           this.tbbRemoveHorizSpace.Enabled = true;
                           this.tbbRemoveVertSpace.Enabled = true;
                           this.tbbSendToBack.Enabled = true;
                           this.tbbSizeToGrid.Enabled = true;
                           break;
                 case toolbarModes.singleControl:
                           this.tbbAlignBottoms.Enabled =false;
                           this.tbbAlignCenters.Enabled = false;
                           this.tbbAlignLefts.Enabled = false;
                           this.tbbAlignMiddles.Enabled = false;
                           this.tbbAlignRights.Enabled = false;
                           this.tbbAlignToGrid.Enabled = true;
                           this.tbbAlignTops.Enabled = false;
                           this.tbbBringToFront.Enabled = true;
                           this.tbbCenterHoriz.Enabled = true;
                           this.tbbCenterVert.Enabled = true;
                           this.tbbDecreaseHorizSpace.Enabled = false;
                           this.tbbDecreaseVertSpace.Enabled = false;
                           this.tbbIncreaseHorizSpace.Enabled = false;
                           this.tbbIncreaseVertSpace.Enabled = false;
                           this.tbbMakeHorizSpaceEqual.Enabled = false;
                           this.tbbMakeSameHeight.Enabled = false;
                           this.tbbMakeSameSize.Enabled = false;
                           this.tbbMakeSameWidth.Enabled = false;
                           this.tbbMakeVertSpaceEqual.Enabled = false;
                           this.tbbRemoveHorizSpace.Enabled = false;
                           this.tbbRemoveVertSpace.Enabled = false;
                           this.tbbSendToBack.Enabled = true;
                           this.tbbSizeToGrid.Enabled = true;
                           break;
                 case toolbarModes.noControls:
                           this.tbbAlignBottoms.Enabled =false;
                           this.tbbAlignCenters.Enabled = false;
                           this.tbbAlignLefts.Enabled = false;
                           this.tbbAlignMiddles.Enabled = false;
                           this.tbbAlignRights.Enabled = false;
                           this.tbbAlignToGrid.Enabled = false;
                           this.tbbAlignTops.Enabled = false;
                           this.tbbBringToFront.Enabled = false;
                           this.tbbCenterHoriz.Enabled = false;
                           this.tbbCenterVert.Enabled = false;
                           this.tbbDecreaseHorizSpace.Enabled = false;
                           this.tbbDecreaseVertSpace.Enabled = false;
                           this.tbbIncreaseHorizSpace.Enabled = false;
                           this.tbbIncreaseVertSpace.Enabled = false;
                           this.tbbMakeHorizSpaceEqual.Enabled = false;
                           this.tbbMakeSameHeight.Enabled = false;
                           this.tbbMakeSameSize.Enabled = false;
                           this.tbbMakeSameWidth.Enabled = false;
                           this.tbbMakeVertSpaceEqual.Enabled = false;
                           this.tbbRemoveHorizSpace.Enabled = false;
                           this.tbbRemoveVertSpace.Enabled = false;
```

```
                             this.tbbSendToBack.Enabled = false;
                             this.tbbSizeToGrid.Enabled = false;
                             break;
          }
          this.ResumeLayout();
}
```

Adding code for the Layout toolbar's executeLayoutAction event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the executeLayoutAction event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the executeLayoutAction event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Private Sub executeLayoutAction(ByVal actionTool As String)
        Select Case actionTool
            Case "BringToFront"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatOrderBringToFront)
            Case "SendToBack"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatOrderSendToBack)
            Case "MakeSameHeight"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSizeSameHeight)
            Case "MakeSameWidth"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSizeSameWidth)
            Case "MakeSameSize"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSizeBoth)
            Case "AlignTops"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignTop)
            Case "AlignBottoms"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignBottom)
            Case "AlignLefts"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignLeft)
            Case "AlignRights"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignRight)
            Case "AlignMiddles"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignMiddle)
            Case "AlignCenters"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatAlignCenter)
            Case "SizeToGrid"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.SnapToGrid)
            Case "MakeHorizSpaceEqual"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceEquallyHorizontal)
            Case "IncreaseHorizSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceIncreaseHorizontal)
            Case "DecreaseHorizSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceDecreaseHorizontal)
            Case "MakeVertSpaceEqual"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceEquallyVertical)
            Case "IncreaseVertSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceIncreaseVertical)
            Case "DecreaseVertSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceDecreaseVertical)
            Case "CenterHoriz"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatCenterHorizontally)
            Case "CenterVert"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatCenterVertically)
            Case "RemoveHorizSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceRemoveHorizontal)
            Case "RemoveVertSpace"
                Me.Designer1.ExecuteAction(DataDynamics._
                            ActiveReports.Design._
                   DesignerAction.FormatSpaceRemoveVertical)
        End Select
End Sub

[C#]

private void executeLayoutAction(string actionTool)
{
         switch(actionTool)
         {
                  case "BringToFront":
                           this.designer1.ExecuteAction(DataDynamics.
                                    ActiveReports.Design.
                                    DesignerAction.FormatOrderBringToFront);
                           break;
                  case "SendToBack":
                           this.designer1.ExecuteAction(DataDynamics.
                                    ActiveReports.Design.
                                    DesignerAction.FormatOrderSendToBack);
                           break;
                  case "MakeSameHeight":
                           this.designer1.ExecuteAction(DataDynamics.
                                    ActiveReports.Design.
```

```
                                        DesignerAction.FormatSizeSameHeight);
                              break;
                    case "MakeSameWidth":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSizeSameWidth);
                              break;
                    case "MakeSameSize":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSizeBoth);
                              break;
                    case "AlignTops":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignTop);
                              break;
                    case "AlignBottoms":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignBottom);
                              break;
                    case "AlignLefts":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignLeft);
                              break;
                    case "AlignRights":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignRight);
                              break;
                    case "AlignMiddles":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignMiddle);
                              break;
                    case "AlignCenters":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatAlignCenter);
                              break;
                    case "SizeToGrid":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.SnapToGrid);
                              break;
                    case "MakeHorizSpaceEqual":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceEqually
                                        Horizontal);
                              break;
                    case "IncreaseHorizSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceIncrease
                                        Horizontal);
                              break;
                    case "DecreaseHorizSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ctiveReports.Design.
                                        DesignerAction.FormatSpaceDecrease
                                        Horizontal);
                              break;
                    case "MakeVertSpaceEqual":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceEqually
                                        Vertical);
                              break;
                    case "IncreaseVertSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceIncrease
                                        Vertical);
                              break;
                    case "DecreaseVertSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceDecrease
                                        Vertical);
                              break;
                    case "CenterHoriz":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatCenter
                                        Horizontally);
                              break;
                    case "CenterVert":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatCenter
                                        Vertically);
                              break;
                    case "RemoveHorizSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceRemove
                                        Horizontal);
                              break;
                    case "RemoveVertSpace":
                              this.designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatSpaceRemove
                                        Vertical);
                              break;
          }
}
```

Adding code for tlbLayout's ButtonClick event

To write the code in Visual Basic

- Double-click on tlbLayout. This creates an event-handling method for tlbLayout's ButtonClick event. Add the following code to the tlbLayout_ButtonClick event.

To write the code in C#

- Double-click on tlbLayout. This creates an event-handling method for tlbLayout's ButtonClick event. Add the following code to the tlbLayout_ButtonClick event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub tlbLayout_ButtonClick(ByVal sender As Object, _
          ByVal e As System.Windows.Forms.ToolBarButtonClickEventArgs)_
          Handles tlbLayout.ButtonClick
        Me.executeLayoutAction(e.Button.Tag.ToString())
End Sub

[C#]

private void tlbLayout_ButtonClick(object sender, System.Windows.Forms.
          ToolBarButtonClickEventArgs e)
```

```
{
         this.executeLayoutAction(e.Button.Tag.ToString());
}
```

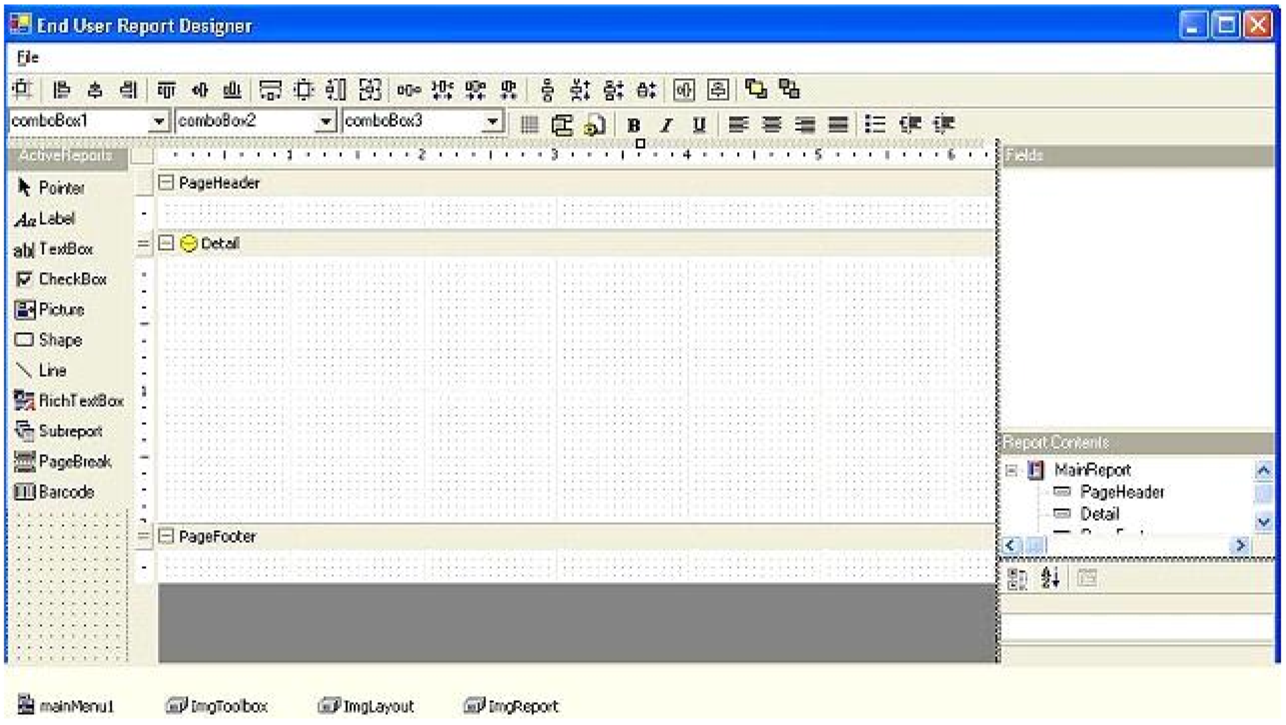
Configuring the Report Toolbar

This walkthrough illustrates how to set up the ActiveReports Toolbox to add to the basic layout of your End-User Report Designer. This walkthrough builds on the walkthroughs "Creating the Basic Layout for an End-User Report Designer", "Configuring the ActiveReports Toolbox" and "Configuring the Layout Toolbar."

This walkthrough is split up into the following activities:

- Adding buttons to the Toolbar collection
- Adding an ImageList to add icons for the Toolbar
- Adding the icon images to the ImageList
- Setting properties for the buttons in the Toolbar collection

When you have completed this walkthrough your layout for End-User Report Designer will look similar to the following.



Adding buttons to the Toolbar collection

To add buttons to the Toolbar collection

1. Click on tlbReport to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. Add seventeen buttons to the collection and click **OK**.

Adding an ImageList to add icons

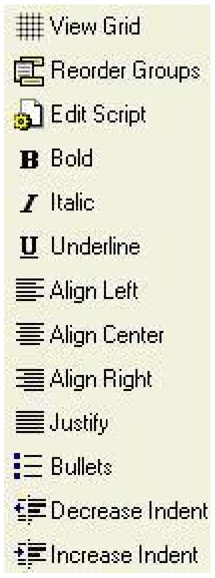
To add an ImageList

1. Click at the top of frmDesigner to select it.
2. Click on the ImageList control in the toolbox and drag it onto the form.
3. Change the name of ImageList to ImgReport.
4. Set the ImageList property for tlbReport to ImgReport.

Adding the icon images to the ImageList

To add the icon images

1. Click on ImgReport at the bottom of frmDesigner to select it.
2. Click on the ellipsis for the Images (collection) property.
3. Add the following icons to the ImgReport:



4. Click **OK** to continue.

Setting properties for the buttons in the Toolbox collection

To set properties for the buttons

1. Click on tlbReport to select it.
2. Click on the ellipsis for the Buttons (collection) property.
3. For each button in the collection, make the following property changes and add the appropriate icons to the ImageIndex.

Button #	Tag	Name	Style	ToolTipText
Button1	(Empty string)	**tbbLine8**	Separator	(Empty string)
Button2	ViewGrid	**tbbViewGrid**	Toggle	ViewGrid
Button3	ReorderGroups	**tbbReorderGroups**	PushButton	ReorderGroups

Button4	EditScript	**tbbEditScript**	PushButton	EditScript
Button5	(Empty string)	**tbbLine9**	Separator	(Empty string)
Button6	Bold	**tbbBold**	PushButton	Bold
Button7	Italic	**tbbItalic**	PushButton	Italic
Button8	Underline	**tbbUnderline**	PushButton	Underline
Button9	(Empty string)	**tbbLine10**	Separator	(Empty string)
Button10	AlignLeft	**tbbAlignLeft**	Toggle	AlignLeft
Button11	Center	**tbbCenter**	Toggle	Center
Button12	AlignRight	**tbbAlignRight**	Toggle	AlignRight
Button13	Justify	**tbbJustify**	Toggle	Justify
Button14	(Empty string)	**tbbLine11**	Separator	(Empty string)
Button15	Bullets	**tbbBullets**	Toggle	Bullets
Button16	DecreaseIndent	**tbbDecreaseIndent**	PushButton	DecreaseIndent
Button17	IncreaseIndent	**tbbIncreaseIndent**	PushButton	IncreaseIndent

4. Click **OK** to continue.
5. Set the Appearance property for tlbReport to Flat.
6. Resize Panel1.

Adding Code for the ComboBoxes

This walkthrough is split up into the following activities:

- Adding code for the ComboBox operations events
- Adding code to adjust text in the ComboBoxes
- Adding code for the cmbFonts SelectIndexChanged event
- Adding code for the cmbFontSize SelectIndexChanged event
- Adding code for the cmbClassName SelectIndexChanged event

Adding code for the ComboBox operations events

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the ComboBox operations events.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the ComboBox operations events.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub fillFonts()
        Me.cmbFonts.Items.Clear()
        Dim fc As New System.Drawing.Text.InstalledFontCollection()
        For i = 0 To fc.Families.Length - 1
            Me.cmbFonts.Items.Add(fc.Families(i).Name)
        Next
        Me.cmbFonts.SelectedIndex = 0
End Sub

Private Sub fillFontSizes()
        cmbFontSize.Items.Clear()
        cmbFontSize.Items.Add(" 8")
        cmbFontSize.Items.Add(" 9")
        cmbFontSize.Items.Add("10")
        cmbFontSize.Items.Add("11")
        cmbFontSize.Items.Add("12")
        cmbFontSize.Items.Add("14")
        cmbFontSize.Items.Add("16")
        cmbFontSize.Items.Add("18")
        cmbFontSize.Items.Add("20")
        cmbFontSize.Items.Add("22")
        cmbFontSize.Items.Add("24")
        cmbFontSize.Items.Add("26")
        cmbFontSize.Items.Add("28")
        cmbFontSize.Items.Add("36")
        cmbFontSize.Items.Add("48")
        cmbFontSize.Items.Add("72")
        Me.cmbFontSize.SelectedIndex = 0
End Sub

Private Sub fillClassName()
        Me.cmbClassName.Items.Clear()
        For i = 0 To Designer1.Report.StyleSheet.Count - 1
            Me.cmbClassName.Items.Add(Designer1.Report.StyleSheet(i).Name)
        Next
        Me.cmbClassName.SelectedIndex = 0
End Sub

[C#]

private void fillFonts()
{
         this.cmbFonts.Items.Clear();
         System.Drawing.Text.FontCollection fc = new
                  System.Drawing.Text.InstalledFontCollection();
         for(int i=0;i<fc.Families.Length;i++)
         {
                  this.cmbFonts.Items.Add(fc.Families[i].Name);
         }
}

private void fillFontSizes()
{
         cmbFontSize.Items.Clear();
         cmbFontSize.Items.Add(" 8");
         cmbFontSize.Items.Add(" 9");
         cmbFontSize.Items.Add("10");
         cmbFontSize.Items.Add("11");
         cmbFontSize.Items.Add("12");
         cmbFontSize.Items.Add("14");
         cmbFontSize.Items.Add("16");
         cmbFontSize.Items.Add("18");
         cmbFontSize.Items.Add("20");
         cmbFontSize.Items.Add("22");
         cmbFontSize.Items.Add("24");
         cmbFontSize.Items.Add("26");
         cmbFontSize.Items.Add("28");
         cmbFontSize.Items.Add("36");
         cmbFontSize.Items.Add("48");
         cmbFontSize.Items.Add("72");
}

private void fillClassName()
```

```
{
         this.cmbClassName.Items.Clear();
         for(int i=0;i<designer1.Report.StyleSheet.Count;i++)
         {
                  this.cmbClassName.Items.Add(designer1.Report.StyleSheet
                           [i].Name);
         }
}
```

Adding code to adjust text in the ComboBoxes

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the events to adjust text in the ComboBoxes.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the events to adjust text in the ComboBoxes.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub setFont()
        For i = 0 To i < Designer1.Selection.Count - 1
            ctl = Designer1.Selection(i).GetType.ToString()
            If ((ctl.IndexOf("TextBox") > 0) Or _
                   (ctl.IndexOf("CheckBox") > 0) Or (ctl.IndexOf_
                   ("Label") > 0)) Then
                Select Case ctl
                    Case "DataDynamics.ActiveReports.TextBox"
                        If cmbFontSize.Text <> "" Then
                            CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports.TextBox)._
                                      Font = New Font(cmbFonts.Text, _
                                      (Long.Parse(cmbFontSize.Text)))
                        Else
                            CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      TextBox).Font = New Font_
                             (cmbFonts.Text, (Long.Parse(Font._
                                      Size.ToString)))
                        End If
                    Case "DataDynamics.ActiveReports.Label"
                        If cmbFontSize.Text <> "" Then
                            CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      Label).Font = New Font_
                             (cmbFonts.Text, (Long.Parse_
                                      (cmbFontSize.Text)))
                        Else
                            CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      Label).Font = New Font_
                             (cmbFonts.Text, (Long.Parse(Font._
                                      Size.ToString)))
                        End If
                    Case "DataDynamics.ActiveReports.CheckBox"
                        If cmbFontSize.Text <> "" Then
                            CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      CheckBox).Font = New Font_
                             (cmbFonts.Text, (Long._
                                      Parse(cmbFontSize.Text)))
                        Else
                            CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      CheckBox).Font = New Font_
                             (cmbFonts.Text, (Long.Parse_
                                      (Font.Size.ToString)))
                        End If
                End Select
            End If
        Next
End Sub

Private Sub setClassName()
        For i = 0 To i < Designer1.Selection.Count - 1
            ctl = Designer1.Selection(i).GetType.ToString()
            If ctl.IndexOf("TextBox") > 0 Or ctl.IndexOf_
                   ("CheckBox") > 0 Or ctl.IndexOf("Label") _
                   > 0 Then
                Select Case ctl
                    Case "DataDynamics.ActiveReports.TextBox"
                        CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      TextBox).ClassName = _
                             cmbClassName.Text
                    Case "DataDynamics.ActiveReports.Label"
                        CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      Label).ClassName = _
                             cmbClassName.Text
                    Case "DataDynamics.ActiveReports.CheckBox"
                        CType(Designer1.Selection(i), _
                                      DataDynamics.ActiveReports._
                                      CheckBox).ClassName = _
                             cmbClassName.Text
                End Select
            End If
        Next
End Sub

[C#]

private void setFont()
{
         for(int i=0;i<designer1.Selection.Count;i++)
                             {
                                      string ctl = designer1.Selection[i].
                                               GetType().ToString();
                                      if((ctl.IndexOf("TextBox")
                                               >0)||(ctl.IndexOf
                                               ("CheckBox") >0)||(ctl.
                                               IndexOf("Label") >0))
                   {
                             switch(ctl)
                             {
                                      case "DataDynamics.ActiveReports.
                                               TextBox":
                                               if(cmbFontSize.Text!="")
                                               {
                                                        ((DataDynamics.Active
                                                                 Reports.TextBox)
                                                                 designer1.
                                                                 Selection[i]).Font
                                                                 = new Font
                                                                 (cmbFonts.
                                                                 Text,(float.
                                                                 Parse(cmb
                                                                 FontSize.Text)));
                                               }
                                               else
                    {
                    ((DataDynamics.ActiveReports.
                        TextBox)designer1.
                        Selection[i]).Font
                        = new Font(cmbFonts.
                        Text,float.Parse
                        (((DataDynamics.
                        ActiveReports.
                        TextBox)designer1.
                        Selection[i]).Font.
                        Size.ToString()));
                    }
                    break;
                case "DataDynamics.ActiveReports.Label":
                    if(cmbFontSize.Text!="")
                    {
                    ((DataDynamics.Active
                        Reports.Label)
                        designer1.Selection
                        [i]).Font = new
                        Font(cmbFonts.Text,
                        (cmbFontSize.Text)));
                    }
                    else
                    {
                    ((DataDynamics.Active
                        Reports.Label)
                        designer1.Selection
                        [i]).Font = new
                        Font(cmbFonts.Text,
                        float.Parse
                        (((DataDynamics.
                        ActiveReports.Label)
                        designer1.Selection
                        [i]).Font.Size.
                        ToString()));
                    }
                    break;
                case "DataDynamics.ActiveReports.
                    CheckBox":
                    if(cmbFontSize.Text!="")
                    {
                    ((DataDynamics.Active
                        Reports.CheckBox)
                        designer1.Selection
                        [i]).Font = new
                        Font(cmbFonts.Text,
                        (float.Parse
                        (cmbFontSize.Text)));
                    }
                    else
                    {
                    ((DataDynamics.Active
                        Reports.CheckBox)
                        designer1.Selection
                        [i]).Font = new
                        Font(cmbFonts.Text,
                        float.Parse
                        (((DataDynamics.
                        ActiveReports.
                        CheckBox)designer1.
                        Selection[i]).
                        Font.Size.
                        ToString()));
                    }
                    break;
            }
        }
    }
}
```

```
private void setClassName()
{
         for(int i=0;i<designer1.Selection.Count;i++)
         {
                  string ctl = designer1.Selection[i].
                           GetType().ToString();
                  if((ctl.IndexOf("TextBox") >0)||(ctl.IndexOf("CheckBox")
                           >0)||(ctl.IndexOf("Label") >0))
                  {
                           switch(ctl)
                           {
                                    case "DataDynamics.ActiveReports.
                                             TextBox":
                                             ((DataDynamics.ActiveReports.
                                                      TextBox)
                                                      designer1.Selection[i]).
                                                               ClassName =
                                                               cmbClassName.Text;
                                             break;
                                    case "DataDynamics.ActiveReports.Label":
                                             ((DataDynamics.ActiveReports.
                                                      Label)designer1.Selection
                                                      [i]).ClassName =
                                                      cmbClassName.Text;
                                             break;
                                    case "DataDynamics.ActiveReprots.
                                             CheckBox":
                                             ((DataDynamics.ActiveReports.
                                                      CheckBox)
                                                      designer1.Selection
                                                      [i]).ClassName =
                                                      cmbClassName.Text;
                                             break;
                           }
                  }
         }
}
```

Adding code for the cmbFonts SelectIndexChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *cmbFonts*. At the top right of the code window, click the drop-down arrow and select *SelectIndexChanged*. This creates an event-handling method for the cmbFonts_SelectIndexChanged event.

To write the code in C#

- Click on the fonts ComboBox cmbFonts. Click on the events icon in the **Properties** window to display available events for the control. Double-click *SelectIndexChanged*. This creates an event-handling method for the cmbFonts_SelectIndexChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub cmbFonts_SelectedIndexChanged(ByVal sender As Object, _
         ByVal e As System.EventArgs) _
         Handles cmbFonts.SelectedIndexChanged
        If cmbFonts.Text <> "" Then
            setFont()
        End If
End Sub

[C#]

private void cmbFonts_SelectedIndexChanged(object sender,
         System.EventArgs e)
{
         if(cmbFonts.Text != "")
         setFont();
}
```

Adding code for the cmbFontSize_SelectIndexChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *cmbFontSize*. At the top right of the code window, click the drop-down arrow and select *SelectIndexChanged*. This creates an event-handling method for the cmbFontSize_SelectIndexChanged event.

To write the code in C#

- Click on the font size ComboBox cmbFontSize. Click on the events icon in the **Properties** window to display available events for the control. Double-click *SelectIndexChanged*. This creates an event-handling method for the cmbFontSize_SelectIndexChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub cmbFontSize_SelectedIndexChanged(ByVal sender As _
         Object, ByVal e As System.EventArgs) _
         Handles cmbFontSize.SelectedIndexChanged
        If cmbFontSize.Text <> "" Then
            setFont()
        End If
End Sub

[C#]

private void cmbFontSize_SelectedIndexChanged(object sender,
         System.EventArgs e)
{
         if(cmbFontSize.Text != "")
         setFont();
}
```

Adding code for the cmbClassName_SelectIndexChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *cmbClassName*. At the top right of the code window, click the drop-down arrow and select *SelectIndexChanged*. This creates an event-handling method for the cmbClassName_SelectIndexChanged event.

To write the code in C#

- Click on the fonts ComboBox cmbClassName. Click on the events icon in the **Properties** window to display available events for the control. Double-click *SelectIndexChanged*. This creates an event-handling method for the cmbClassName_SelectIndexChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub cmbClassName_SelectedIndexChanged(ByVal sender As _
         Object, ByVal e As System.EventArgs) _
         Handles cmbClassName.SelectedIndexChanged
        If cmbClassName.Text <> "" Then
           setClassName()
        End If
End Sub

[C#]

private void cmbClassName_SelectedIndexChanged(object sender,
         System.EventArgs e)
{
         if(cmbClassName.Text != "")
         setClassName();
}
```

Adding Code for the Report Toolbar

This walkthrough is split up into the following activities:

- Adding code for the Report toolbar's SelectionChanged event
- Adding code for DesignerAction to be returned
- Adding code for the Report toolbar's SetStatus event
- Adding code for the Designer1 StatusChanged event
- Adding code for the Report toolbar's executeLayoutAction event
- Adding code for the Report toolbar's ButtonClick event

Adding code for the Report toolbar's SelectionChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the selChangeReportToolbar event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the selChangeReportToolbar event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

 Private Sub selChangeReportToolbar()
```

```
        Dim m_arrayFont As New ArrayList()
        Dim m_arrayFontSize As New ArrayList()
        Dim m_arrayClassName As New ArrayList()
        Dim sFont As String
        Dim sFontSize As String
        Dim sClassName As String
        sFont = ""
        sFontSize = ""
        sClassName = ""
        Dim ctl As String
        cnt = Designer1.Selection.Count
        If cnt = 0 Then
            Exit Sub
        End If
        For i = 0 To cnt - 1
            ctl = Designer1.Selection(i).GetType().ToString
            If ((ctl.IndexOf("TextBox") > 0) Or _
                   (ctl.IndexOf("CheckBox") > 0) _
                   Or (ctl.IndexOf("Label") > 0)) Then
                Select Case ctl
                    Case "DataDynamics.ActiveReports.TextBox"
                        sFont = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      TextBox).Font.Name.ToString()
                        sFontSize = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      TextBox).Font.Size.ToString()
                        sClassName = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      TextBox).ClassName.ToString()
                        If m_arrayFont.Contains(sFont) = False Then
                            m_arrayFont.Add(sFont)
                            If m_arrayFontSize.Contains(sFontSize) _
                                      = False Then
                                m_arrayFontSize.Add(sFontSize)
                                If m_arrayClassName.Contains_
                                                (sClassName) = False Then
                                    m_arrayClassName.Add(sClassName)
                                End If
                            End If
                        End If
                    Case "DataDynamics.ActiveReports.Label"
                        sFont = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      Label).Font.Name.ToString()
                        sFontSize = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      Label).Font.Size.ToString()
                        sClassName = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      Label).ClassName.ToString()
                        If m_arrayFont.Contains(sFont) = False Then
                            m_arrayFont.Add(sFont)
                            If m_arrayFontSize.Contains_
                                                (sFontSize) = False Then
                                m_arrayFontSize.Add(sFontSize)
                                If m_arrayClassName.Contains_
                                                (sClassName) = False Then
                                    m_arrayClassName.Add(sClassName)
                                End If
                            End If
                        End If
                    Case "DataDynamics.ActiveReports.CheckBox"
                        sFont = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      CheckBox).Font.Name
                        sFontSize = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      CheckBox).Font.Size
                        sClassName = CType(Designer1.Selection(i),_
                                      DataDynamics.ActiveReports._
                                      CheckBox).ClassName
                        If m_arrayFont.Contains(sFont) = False Then
                            m_arrayFont.Add(sFont)
                            If m_arrayFontSize.Contains(sFontSize)_
                                       = False Then
                                m_arrayFontSize.Add(sFontSize)
                                If m_arrayClassName._
                                               Contains(sClassName) = False Then
                                    m_arrayClassName.Add(sClassName)
                                End If
                            End If
                        End If
                End Select
            End If
        Next
        If ((m_arrayFont.Count >= 1) AndAlso (cnt > 1)) Then
            cmbFonts.Text = ""
            cmbFonts.Enabled = True
        ElseIf (m_arrayFont.Count = 0) Then
            cmbFonts.Text = ""
            cmbFonts.Enabled = False
        ElseIf (m_arrayFont.Count = 1) Then
            cmbFonts.Text = m_arrayFont(0).ToString()
            cmbFonts.Enabled = True
        End If
        If ((m_arrayFont.Count >= 1) AndAlso (cnt > 1)) Then
            cmbFontSize.Text = ""
            cmbFontSize.Enabled = True
        ElseIf (m_arrayFontSize.Count = 0) Then
            cmbFontSize.Text = ""
            cmbFontSize.Enabled = False
        ElseIf (m_arrayFontSize.Count = 1) Then
            cmbFontSize.Text = m_arrayFontSize(0).ToString()
            cmbFontSize.Enabled = True
        End If
        If ((m_arrayFont.Count >= 1) AndAlso (cnt > 1)) Then
            cmbClassName.Text = ""
            cmbClassName.Enabled = True
        ElseIf (m_arrayClassName.Count = 0) Then
            cmbClassName.Text = ""
            cmbClassName.Enabled = False
        ElseIf (m_arrayClassName.Count = 1) Then
            cmbClassName.Text = m_arrayClassName(0).ToString()
            cmbClassName.Enabled = True
        End If
End Sub

[C#]

private void selChangeReportToolbar()
{
         ArrayList m_arrayFont=new ArrayList();
         ArrayList m_arrayFontSize = new ArrayList();
         ArrayList m_arrayClassName = new ArrayList();
         string sFont=null;
         string sFontSize=null;
         string sClassName=null;
         int cnt = this.designer1.Selection.Count;
         for(int i=0;i<cnt;i++)
                            {
                                      string ctl = designer1.Selection[i].
                                               GetType().ToString();
                                      if((ctl.IndexOf("TextBox") >0)||
                                      (ctl.IndexOf("CheckBox") >0)||(ctl.
                                      IndexOf("Label") >0))
                   {
                            switch(ctl)
                            {
                                      case "DataDynamics.ActiveReports.
                                               TextBox":
                                               sFont =((DataDynamics.
                                        ActiveReports.
                                        TextBox)designer1.
                                        Selection[i]).
                                        Font.Name.ToString();
                              sFontSize =((DataDynamics.
                                        ActiveReports.
                                        TextBox)designer1.
                                        Selection[i]).
                                        Font.Size.ToString();
                              sClassName =((DataDynamics.
                                        ActiveReports.
                                        TextBox)designer1.
                                        Selection[i]).
                                        ClassName.ToString();
                              if(m_arrayFont.Contains(sFont)==false)
                                        m_arrayFont.Add(sFont);
                              if(m_arrayFontSize.Contains
                                        (sFontSize)==false)
                                        m_arrayFontSize.
                                        Add(sFontSize);
                              if(m_arrayClassName.Contains
                                        (sClassName)==false)
                                        m_arrayClassName.
                                        Add(sClassName);
                              break;
                    case "DataDynamics.ActiveReports.Label":
                              sFont =((DataDynamics.
                                        ActiveReports.
                                        Label)designer1.
                                        Selection[i]).
                                        Font.Name.ToString();
                              sFontSize =((DataDynamics.
                                        ActiveReports.Label)
                                        designer1.
                                        Selection[i]).
                                        Font.Size.ToString();
                              sClassName =((DataDynamics.
                                        ActiveReports.
                                        Label)designer1.
                                        Selection[i]).
                                        ClassName.ToString();
                              if(m_arrayFont.Contains(sFont)
                                        ==false)
                                        m_arrayFont.Add(sFont);
                              if(m_arrayFontSize.Contains
                                        (sFontSize)==false)
                                        m_arrayFontSize.Add
                                                  (sFontSize);
                              if(m_arrayClassName.Contains
                                        (sClassName)==false)
                                        m_arrayClassName.Add
                                                  (sClassName);
                              break;
                    case "DataDynamics.ActiveReports.
                              CheckBox":
                              sFont =((DataDynamics.
                                        ActiveReports.
                                        CheckBox)designer1.
                                        Selection[i]).
                                        Font.Name.ToString();
                              sFontSize =((DataDynamics.
                                        ActiveReports.
                                        CheckBox)designer1.
                                        Selection[i]).
                                        Font.Size.ToString();
                              sClassName =((DataDynamics.
                                        ActiveReports.
                                        CheckBox)designer1.
                                        Selection[i]).
                                        ClassName.ToString();
                              if(m_arrayFont.Contains
                                                  (sFont)==false)
                                                  m_arrayFont.Add(sFont);
                                        if(m_arrayFontSize.Contains
                                                  (sFontSize)==false)
                                                  m_arrayFontSize.
                                                            Add(sFontSize);
                                        if(m_arrayClassName.Contains
                                                  (sClassName)==false)
                                                  m_arrayClassName.
                                                            Add(sClassName);
                                        break;
                              }
                    }
                    if((m_arrayFont.Count>=1) && (cnt>1))
                    {
                              cmbFonts.Text = "";
                              cmbFonts.Enabled =true;
                    }
                    else if(m_arrayFont.Count==0)
                    {
                              cmbFonts.Text = "";
                              cmbFonts.Enabled = false;
                    }
                    else if(m_arrayFont.Count ==1)
                    {
                              cmbFonts.Text = m_arrayFont[0].ToString();
                              cmbFonts.Enabled =true;
                    }
                    if((m_arrayFont.Count>=1) && (cnt>1))
                    {
                              cmbFontSize.Text = "";
                              cmbFontSize.Enabled = true;
                    }
                    else if(m_arrayFontSize.Count ==0)
                    {
                              cmbFontSize.Text = "";
                              cmbFontSize.Enabled = false;
                    }
                    else if(m_arrayFontSize.Count==1)
                    {
                              cmbFontSize.Text = m_arrayFontSize[0].ToString();
                              cmbFontSize.Enabled = true;
                    }

                    if((m_arrayFont.Count>=1) && (cnt>1))
                    {
                              cmbClassName.Text = "";
                              cmbClassName.Enabled = true;
                    }
                    else if(m_arrayClassName.Count==0)
                    {
                              cmbClassName.Text = "";
                              cmbClassName.Enabled = false;
                    }
                    else if(m_arrayClassName.Count ==1)
                    {
                              cmbClassName.Text = m_arrayClassName
                                        [0].ToString();
                              cmbClassName.Enabled = true;
                    }
          }
}
```

Adding code for the DesignerAction to be returned

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the getActionFromString function.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the getActionFromString function.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Function getActionFromString(ByVal action As String) _
          As DataDynamics.ActiveReports.Design.DesignerAction
        Dim m_action As New DataDynamics.ActiveReports.Design._
                   DesignerAction()
        Select Case (action)
            Case "Bold"
                m_action = DesignerAction.FormatFontBold
            Case "Italic"
                m_action = DesignerAction.FormatFontItalic
            Case "Underline"
                m_action = DesignerAction.FormatFontUnderline
            Case "Bullets"
                m_action = DesignerAction.FormatRTFBullets
            Case "IncreaseIndent"
                m_action = DesignerAction.FormatRTFIndent
            Case "DecreaseIndent"
                m_action = DesignerAction.FormatRTFOutdent
            Case "EditScript"
                m_action = DesignerAction.EditScript
            Case "ReorderGroups"
                m_action = DesignerAction.ReorderGroups
            Case "ViewGrid"
                m_action = DesignerAction.ViewGrid
            Case "AlignLeft"
                m_action = DesignerAction.FormatTextAlignLeft
            Case "Center"
                m_action = DesignerAction.FormatTextAlignCenter
            Case "AlignRight"
                m_action = DesignerAction.FormatTextAlignRight
            Case "Justify"
                m_action = DesignerAction.FormatTextAlignJustify
        End Select
        Return m_action
End Function

[C#]

private DataDynamics.ActiveReports.Design.DesignerAction
          getActionFromString(string action)
{
          switch(action)
          {
                    case "Bold":
                              result =DataDynamics.ActiveReports.Design.
                                        DesignerAction.FormatFontBold;
                              break;
                    case "Italic":
                              result = DataDynamics.ActiveReports.Design.
                                        DesignerAction.FormatFontItalic;
                              break;
                    case "Underline":
                              result = DataDynamics.ActiveReports.Design.
                                        DesignerAction.FormatFontUnderline;
                              break;
                   case "Bullets":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatRTFBullets;
                            break;
                   case "IncreaseIndent":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatRTFIndent;
                            break;
                   case "DecreaseIndent":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatRTFOutdent;
                            break;
                   case "EditScript":
                            result =DataDynamics.ActiveReports.Design.
                                     DesignerAction.EditScript;
                            break;
                   case "ReorderGroups":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.ReorderGroups;
                            break;
                   case "ViewGrid":
                            result=DataDynamics.ActiveReports.Design.DesignerAction.
                   ViewGrid;
                            break;
                   case "AlignLeft":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatTextAlignLeft;
                            break;
                   case "Center":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatTextAlignCenter;
                            break;
                   case "AlignRight":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatTextAlignRight;
                            break;
                   case "Justify":
                            result = DataDynamics.ActiveReports.Design.
                                     DesignerAction.FormatTextAlignJustify;
                            break;
          }
          return result;
}
```

Adding code for the Report toolbar's setStatus event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the setStatus event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the setStatus event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub setStatus(ByVal action As String, ByVal toolbutton _
          As System.Windows.Forms.ToolBarButton)
        Dim m_action As New DataDynamics.ActiveReports._
                   Design.DesignerAction()
        m_action = getActionFromString(action)
        toolbutton.Enabled = Designer1.QueryActionEnabled(m_action)
        toolbutton.Pushed = Designer1.QueryActionChecked(m_action)
End Sub

[C#]

private void setStatus(string action,System.Windows.Forms.
         ToolBarButton toolButton)
{
         DataDynamics.ActiveReports.Design.DesignerAction
                  m_action = getActionFromString(action);
         toolButton.Enabled = designer1.QueryActionEnabled(m_action);
         toolButton.Pushed = designer1.QueryActionChecked(m_action);
}
```

Adding code to the Designer1_StatusChanged event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. At the top left of the code view for frmDesigner, click the drop-down arrow and select *designer1*. At the top right of the code window, click the drop-down arrow and select *StatusChanged*. This creates an event-handling method for the Designer1_StatusChanged event.

To write the code in C#

- Click inside of the designer to select Designer1. Click on the events icon in the **Properties** window to display available events for the section. Double-click *StatusChanged*. This creates an event-handling method for the Designer1_StatusChanged event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub Designer1_StatusChanged(ByVal sender As Object, _
         ByVal e As System.EventArgs) _
         Handles Designer1.StatusChanged
        For i = 0 To Me.tlbReport.Buttons.Count - 1
            Dim tb As New System.Windows.Forms.ToolBarButton()
            tb = Me.tlbReport.Buttons(i)
            If tb.Tag Is Nothing = False Then
                setStatus(tb.Tag, tb)
            End If
        Next
End Sub

[C#]

private void designer1_StatusChanged(object sender, System.EventArgs e)
{
         for(int i=0;i<this.tlbReport.Buttons.Count;i++)
         {
                  System.Windows.Forms.ToolBarButton tb =
                           this.tlbReport.Buttons[i];
                  if(tb.Tag != null)
                  {
                           setStatus (tb.Tag.ToString(),tb);
                  }
         }
}
```

Adding code for the Report toolbar's executeReportAction event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the executeReportAction event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the executeReportAction event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub executeReportAction(ByVal action As String, ByVal _
         e As System.Windows.Forms._
         ToolBarButtonClickEventArgs)
        Try
            Select Case action
                Case "Bold"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatFontBold)
                Case "Italic"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatFontItalic)
                Case "Underline"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatFontUnderline)
                Case "AlignRight"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatTextAlignRight)
                Case "Center"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatTextAlignCenter)
                Case "AlignLeft"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatTextAlignLeft)
                Case "Justify"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatTextAlignJustify)
                Case "Bullets"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatRTFBullets)
                Case "DecreaseIndent"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatRTFOutdent)
                Case "IncreaseIndent"
                    Designer1.ExecuteAction_
                            (DesignerAction.FormatRTFIndent)
            End Select
        Catch ex As Exception
            MessageBox.Show(Me, ex.Message, "Action _
                   Failed", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
End Sub

[C#]

private void executeReportAction(string action)
{
         switch(action)
```

```
          {
                    case "Bold":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.DesignerAction.
                                        FormatFontBold);
                              break;
                    case "Italic":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatFontItalic);
                              break;
                    case "Underline":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatFontUnderline);
                              break;
                    case "Bullets":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatRTFBullets);
                              break;
                    case "IncreaseIndent":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatRTFIndent);
                              break;
                    case "DecreaseIndent":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatRTFOutdent);
                              break;
                    case "EditScript":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.EditScript);
                              break;
                    case "ReorderGroups":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.ReorderGroups);
                              break;
                    case "ViewGrid":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.ViewGrid);
                              break;
                    case "AlignLeft":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatTextAlignLeft);
                              break;
                    case "Center":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatTextAlignCenter);
                              break;
                    case "AlignRight":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatTextAlignRight);
                              break;
                    case "Justify":
                              designer1.ExecuteAction(DataDynamics.
                                        ActiveReports.Design.
                                        DesignerAction.FormatTextAlignJustify);
                              break;
          }
}
```

Adding code for tlbReport's ButtonClick event

To write the code in Visual Basic

- Double-click on tlbReport. This creates an event-handling method for tlbReport's ButtonClick event. Add the following code to the tlbReport_ButtonClick event.

To write the code in C#

- Double-click on tlbReport. This creates an event-handling method for tlbReport's ButtonClick event. Add the following code to the tlbReport_ButtonClick event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub tlbReport_ButtonClick(ByVal sender As Object, ByVal _
         e As System.Windows.Forms.ToolBarButton_
         ClickEventArgs) Handles tlbReport.ButtonClick
        executeReportAction(e.Button.Tag.ToString(), e)
End Sub

[C#]

private void tlbReport_ButtonClick(object sender, System.Windows.
         Forms.ToolBarButtonClickEventArgs e)
{
         executeReportAction(e.Button.Tag.ToString());
}
```

Adding a Viewer Control for the End-User Report Designer

This walkthrough is split up into the following activities:

- Adding a new Windows Form and ActiveReports WinForm viewer control
- Adding code for the viewer's setReport event
- Adding code for the viewer's showReport event
- Adding code for frmPrintPreview Load event

Adding a new Windows Form and ActiveReports viewer control

1. To add a new form and viewer control
2. Add a new "Windows Form" to your Visual Studio project.
3. Change the name of Form1 to frmPrintPreview.
4. Click on the ActiveReports viewer control in the appropriate toolbox and drag it onto Form1.
5. Set the viewer control's Dock property to Fill.

Adding code for the viewer's SetReport event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the SetReport event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the SetReport event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Public Sub SetReport(ByRef report As DataDynamics._
         ActiveReports.ActiveReport)
        Try
            m_rpt = report
        Catch ex As Exception
            MessageBox.Show("Error: " + ex.Message + "\n" + _
                   "Stack:\n" + ex.StackTrace + "\n" + ex.Source)
        End Try
End Sub

[C#]

public void SetReport(DataDynamics.ActiveReports.ActiveReport report)
{
         m_rpt = report;
}
```

Adding code for the viewer's showReport event

To write the code in Visual Basic

- Right-click in any section of frmDesigner, and click on **View Code** to display the code view for the Windows Form. Add the following code to create the showReport event.

To write the code in C#

- Double-click on frmDesigner to see the code view for the Windows form. Add the following code to create the showReport event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub showReport(ByRef reportObject As DataDynamics._
         ActiveReports.ActiveReport)
        Try
            Dim rpt As New DataDynamics.ActiveReports.ActiveReport()
            rpt = reportObject
            Me.Viewer1().Document = rpt.Document
            rpt.Run()
        Catch ex As Exception
            MessageBox.Show("Error:\n " + ex.Message + _
                   "\nStack:\n" + ex.StackTrace + "\nSource:\n" _
         `         + ex.Source)
        End Try
End Sub

[C#]

private void showReport(DataDynamics.ActiveReports.ActiveReport report)
{
         try
         {
                  DataDynamics.ActiveReports.ActiveReport rpt =
                           (DataDynamics.ActiveReports.
                           ActiveReport)report;
                  viewer1.Document = rpt.Document;
```

```
                   rpt.Run();
         }
         catch(Exception ex)
         {
                   MessageBox.Show("Error:\n " + ex.Message +
                            "\nStack:\n"+ ex.StackTrace +
                            "\nSource:\n" + ex.Source);
         }
}
```

Adding code for the frmPrintPreview_Load event

To write the code in Visual Basic

- Right-click on frmPrintPreview, and click on **View Code** to display the code view for the form. At the top left of the code view for frmPrintPreview, click the drop-down arrow and select *(Base Class Events)*. At the top right of the code window, click the drop-down arrow and select *Load*. This creates an event-handling method for the frmPrintPreview_Load event.

To write the code in C#

- Click on the blue section at the top of frmPrintPreview to select the form. Click on the events icon in the **Properties** window to display available events for frmPrintPreview. Double-click *Load*. This creates an event-handling method for the frmPrintPreview_Load event.

The following example shows what the code for the method looks like:

```
[Visual Basic]

Private Sub frmViewer_Load(ByVal sender As System.Object, ByVal _
         e As System.EventArgs) Handles MyBase.Load
        If m_rpt Is Nothing = False Then
            Me.showReport(m_rpt)
        End If
End Sub

[C#]

private void frmViewer_Load(object sender, System.EventArgs e)
{
         if(m_rpt != null)
         {
                   this.showReport(m_rpt);
         }
}
```

Deploying ActiveReports Web Applications

With ActiveReports for .NET, Web applications can be set up for deployment by including the ActiveReports deployment .msm file in your Visual Studio deployment project.

This walkthrough illustrates how to create a deployment project in ActiveReports for a web application.

This walkthrough is split up into the following activities:

- Adding an installer project to an existing ActiveReports web project
- Adding the ActiveReports .msm file
- Adding the ActiveReports application to the installer
- Deploying the installer application to a web server on your computer

Adding an installer project to an existing ActiveReports project

To add the installer project

1. Open an existing ActiveReports project or create a new report.
2. On the **Build** menu, click "Build [your ActiveReports web project name]" to build your report project.
3. On the **File** menu, select **Add Project** and click on **New Project...**
4. Under Project Types in the Add New Project dialog, select Setup and Deployment Projects.
5. In the **Templates** window, select **Web Setup Project,** rename the file and click **OK**.
6. Select the Installer project in Solution Explorer. In the Properties window, select the **ProductName** property and type in the name of your file.

   Note The **ProductName** property determines the name that will be displayed for the application in folder names and in the **Add/Remove Programs** dialog box.

Adding the ActiveReports .msm file

To add the ActiveReports .msm file

1. Right-click on the Installer project in Solution Explorer.
2. Click on **Add** and then click **Merge Module...**
3. Open the Deployment folder where ActiveReports for .NET is installed (e.g. c:\\program files\Data Dynamics\ActiveReports.NET\Deployment).
4. Click on "ActiveReportsDistrib.msm" to select it and click **Open**.
5. This adds all of the ActiveReports distributed assemblies to your web project.

   Note Since the Setup and Deployment project will automatically detect and add any assembly dependencies to your project and the .msm file adds all ActiveReports assemblies, you will need to exclude any duplicate ActiveReports DLLs from the "Detected Dependencies" folder in the Solution Explorer window.

Adding the ActiveReports application to the installer

To add the ActiveReports application

1. Select the Installer project in Solution Explorer.
2. In the File System Editor, choose the Web Application folder.
3. On the Action menu, select Add, Project Output...
4. In the **Add Project Output Group** dialog, choose your ActiveReports project name from the drop-down list.
5. Select "Primary Output" and "Content Files" from the list and click **OK**.
6. Select the Web Application folder. In the **Properties** window, set the **VirtualDirectory** property to "xyz".
7. In the **Properties** window, set the **DefaultDocument** property to "WebForm1.aspx".
8. On the **Build** menu, click "Build [your Installer project name]" to build your Installer project.

Deploying the installer application to a web server on your computer

To deploy the installer application

1. In Solution Explorer, select the web Installer project.
2. On the **Project** menu, click **Install**.
3. To access the web application that was deployed, start Internet Explorer and enter the URL: http://localhost/xyz.

HTTP Handlers

ActiveReports provides HTTPHandler components that, upon configuration, allow ASP.NET to automatically process reports that have been placed into an ASP.NET web site folder. ActiveReports HTTPHandler components enable easily deployable reports in both HTML and PDF file formats. ActiveReports includes a simple configuration utility to properly register the HTTPHandler components with IIS and ASP.NET.

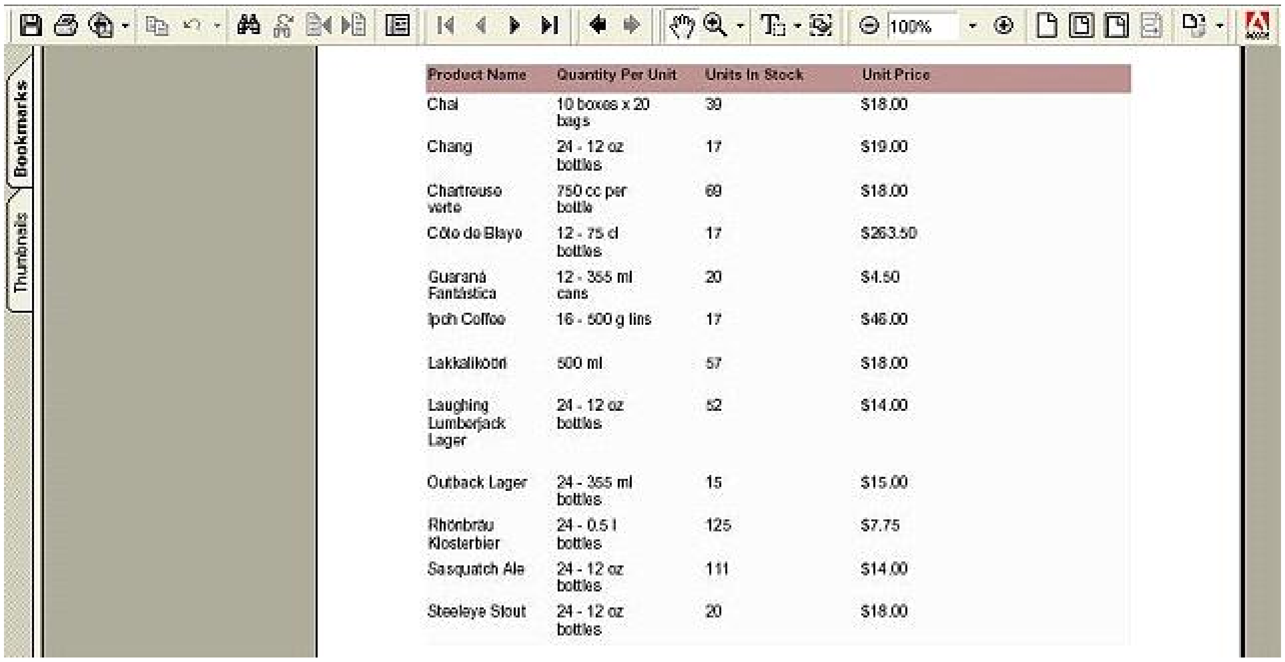
This walkthrough illustrates how to create a simple Web application and set the HTTPHandler to output report information in PDF format.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to an ASP.NET Web application
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding code to the Web.config file to enable HTTP handlers
- Adding a link to the Web Form to enable viewing in PDF format

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb). You must also have access to Internet Information Services either from your computer or from the server. You must also run the "Configure Web Sample" option from the Data Dynamics ActiveReports for .NET program menu from your Windows Start button.

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product Name	Quantity Per Unit	Units In Stock	Unit Price
Chai	10 boxes x 20 bags	39	$18.00
Chang	24 - 12 oz bottles	17	$19.00
Chartreuse verte	750 cc per bottle	69	$18.00
Côte de Blaye	12 - 75 cl bottles	17	$263.50
Guaraná Fantástica	12 - 355 ml cans	20	$4.50
Ipoh Coffee	16 - 500 g tins	17	$46.00
Lakkalikööri	500 ml	57	$18.00
Laughing Lumberjack Lager	24 - 12 oz bottles	52	$14.00
Outback Lager	24 - 355 ml bottles	15	$15.00
Rhönbräu Klosterbier	24 - 0.5 l bottles	125	$7.75
Sasquatch Ale	24 - 12 oz bottles	111	$14.00
Steeleye Stout	24 - 12 oz bottles	20	$18.00

Adding an ActiveReport to an ASP.NET Web application

To add an ActiveReport to your project

1. Open a new ASP.NET Web application in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptHTTPHandlers.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products ORDER BY categoryID, productname".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptWebView.
2. Make the following changes to the group header:

- o Change the name to ghProducts
- o Change the DataField property to CategoryID
- o Change the GroupKeepTogether property to First Detail
- o Change the KeepTogether property to True

3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductName**	Product Name	0, 0
Label	**lblQuantityPerUnit**	Quantity Per Unit	1.1875, 0
Label	**lblInStock**	In Stock	2.5625, 0
Label	**lblUnitPrice**	Unit Price	4, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Format
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	QuantityPerUnit	**txtQuantityPerUnit**	Quantity Per Unit	1.1875, 0	(Empty string)
TextBox	UnitsInStock	**txtInStock**	Units In Stock	2.5625, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4, 0	Currency

Adding code to the Web.config file to enable HTTPHandler

To add code to the Web.config file

1. Double-click the Web.config file in the Solution Explorer window.
2. In the XML view, add the following code in the Globalization section at the end:
3.

```
<httpHandlers>
         <!--
         ********** ActiveReports HttpHandler Configuration **********
         -->
         <add verb="*" path="*.rpx"type="DataDynamics.ActiveReports.
                  Web.Handlers.RpxHandler, ActiveReports.Web,
                  Version=3.0.0.1468, Culture=neutral,
                  PublicKeyToken=d46dd3fabd95c9d1" />

         <add verb="*" path="*.ActiveReport" type="DataDynamics.
                  ActiveReports.Web.Handlers.Compiled
                  ReportHandler, ActiveReports.Web,
                  Version=3.0.0.1468, Culture=neutral,
                  PublicKeyToken=d46dd3fabd95c9d1" />

         <add verb="*" path="*.ArCacheItem" type="DataDynamics.
                  ActiveReports.Web.Handlers.WebCacheAccessHandler,
                  ActiveReports.Web, Version=3.0.0.1468, Culture=
                  neutral, PublicKeyToken=d46dd3fabd95c9d1" />
</httpHandlers>
```

Note The version number will need to be updated to reflect the current version of ActiveReports installed on your computer.

Adding a link to the Web Form

To add a link to the Web Form

1. In the HTML view of the Web Form, add the following HTML code:

```
<a href="rptHTTPHandlers.rpx?OutputFormat=pdf">WebApp.<span
style="COLOR:red">rpx</span>/
<span style="COLOR: green">
</span>OutputFormat=pdf<span style="COLOR:red"></span></a>
```

2. Press F5 to run the program.
3. Click the link on the web form to convert the report preview to PDF.

Web Viewer Control

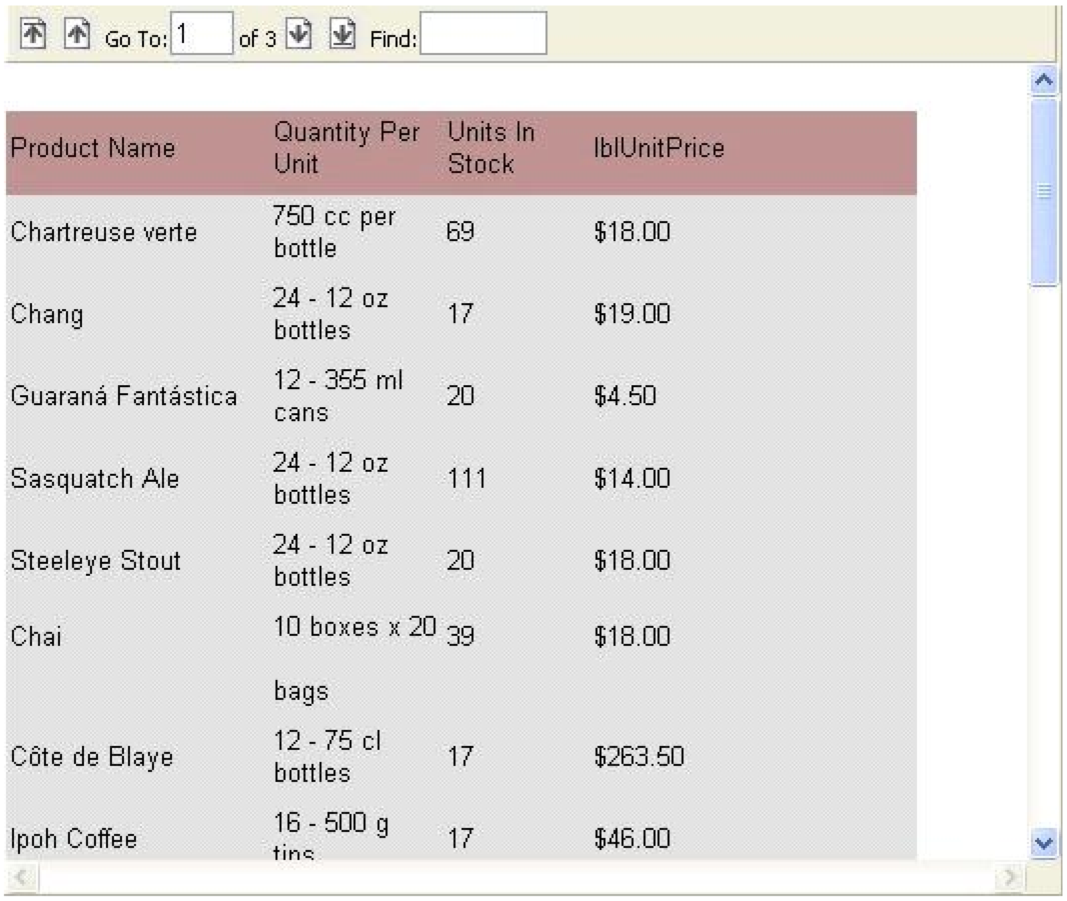
The ActiveReports WebControl allows you to easily publish simple reports to the web for viewing in the browser. The client machine will not require ActiveReports or ASP.NET to be installed. The WebControl also takes advantage of a report queuing technology to ensure the reports are executed and outputted efficiently. To use the WebControl you will select an ActiveReport using the Report property of the WebControl in the property list and set the ViewerType property to the viewer of your choice. Alternatively, you can set the Report property programmatically to a new instance of an ActiveReport class.

This walkthrough is split up into the following activities:

- Adding an ActiveReport to an ASP.NET Web application
- Connecting the report to a data source
- Adding controls to the report to contain data
- Adding the ActiveReports Web Viewer control to the Web Form

To complete the walkthrough, you must have access to the NorthWind database (NWind.mdb). You must also have access to Internet Information Services either from your computer or from the server. You must also run the "Configure Web Sample" option from the Data Dynamics ActiveReports for .NET program menu from your Windows Start button.

When you have completed this walkthrough, you will have a report that looks similar to the following.



Product Name	Quantity Per Unit	Units In Stock	lblUnitPrice
Chartreuse verte	750 cc per bottle	69	$18.00
Chang	24 - 12 oz bottles	17	$19.00
Guaraná Fantástica	12 - 355 ml cans	20	$4.50
Sasquatch Ale	24 - 12 oz bottles	111	$14.00
Steeleye Stout	24 - 12 oz bottles	20	$18.00
Chai	10 boxes x 20 bags	39	$18.00
Côte de Blaye	12 - 75 cl bottles	17	$263.50
Ipoh Coffee	16 - 500 g tins	17	$46.00

Adding an ActiveReport to an ASP.NET Web application

To add an ActiveReport to your project

1. Open a new ASP.NET Web application in Visual Studio.
2. Click on **Project > Add New Item**.
3. Select **ActiveReports file** and rename the file rptWebView.
4. Click **Open**.

Connecting the report to a data source

To connect the report to a data source

1. Click on the yellow report DataSource icon in the Detail section. This brings up the report DataSource dialog box.
2. Click on **Build...**
3. Select Microsoft Jet 4.0 OLE DB Provider and click **Next >>**.
4. Click on the ellipsis to browse for the access path to NWind.mdb. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to continue.
6. In the Query field, type "Select * from products ORDER BY categoryID, productname".
7. Click **OK** to return to the report design surface.

Adding controls to the report to contain data

To add controls to the report

1. Add a GroupHeader/Footer section to rptWebView.
2. Make the following changes to the group header:
   - Change the name to ghProducts
   - Change the DataField property to CategoryID
   - Change the GroupKeepTogether property to First Detail
   - Change the KeepTogether property to True
3. Add the following controls to the GroupHeader section:

Control	Name	Text/Caption	Location
Label	**lblProductName**	Product Name	0, 0
Label	**lblQuantityPerUnit**	Quantity Per Unit	1.1875, 0
Label	**lblInStock**	In Stock	2.5625, 0
Label	**lblUnitPrice**	Unit Price	4, 0

4. Add the following controls to the Detail section:

Control	DataField	Name	Text/Caption	Location	Output Forma
TextBox	ProductName	**txtProductName**	Product Name	0, 0	(Empty string)
TextBox	QuantityPerUnit	**txtQuantityPerUnit**	Quantity Per Unit	1.1875, 0	(Empty string)
TextBox	UnitsInStock	**txtInStock**	Units In Stock	2.5625, 0	(Empty string)
TextBox	UnitPrice	**txtUnitPrice**	Unit Price	4, 0	Currency

Adding the ActiveReports Web Viewer control to the Web Form

To add the web viewer control

1. Click on the ActiveReportViewer control in the appropriate toolbox and drag it onto WebForm1.
2. Adjust the size according to your needs.
3. Change the Report property to rptWebView.
4. Make sure the ViewerType property is set to HtmlViewer.

   Note To view the report in PDF format, change the ViewerType property to AcrobatReader. To use the ActiveX Viewer, change the ViewerType property to ActiveXViewer and paste the ActiveX viewer .cab file in your project folder (for help with this, see "Using ActiveX Viewer Control on the Web").