

ActiveReports 6

This is the help file for GrapeCity ActiveReports version 6, reporting software for use in Visual Studio 2005, 2008 and 2010.

In This Documentation

ActiveReports User Guide

The User Guide has many getting started topics and how-to topics with code samples to copy and paste.

Class Library (on-line documentation)

This is the API documentation with topics for all of the public members of each assembly included with ActiveReports.

ActiveReports User Guide

ActiveReports 6 is a fully integrated Visual Studio component which combines the features of the Visual Studio .NET programming languages with user-friendly controls to provide a powerful report designer.

In This Documentation

Introducing ActiveReports 6

Find out what's new in ActiveReports 6, learn which features are freed from the evaluation banner with the Standard and Professional Edition licenses, and find copyright and license information.

Installation

View requirements for installation of ActiveReports 6, learn what files are installed and how to verify your installation, and find installation troubleshooting tips.

License Your ActiveReports

This topic walks you through how to license your machine and how to add licensing to any projects created during your evaluation.

Upgrading Reports

Use this section to guide you through the upgrade path from previous ActiveReports versions, and to learn to convert Microsoft Access reports to ActiveReports.

Getting Started

This section serves as a guide to help you to get started, with topics on adding controls to Visual Studio, adding reports to projects, and understanding the designer interface. You will also find topics on the report viewer and on using ActiveReports on the Web.

Concepts

This section explains important concepts to help you to understand what to expect from ActiveReports and how it works.

How To

Here you will find topics that guide you through specific tasks that you may want to perform with ActiveReports.

Samples and Walkthroughs

Find out how to use various features of ActiveReports using included sample projects and step-by-step guides to create new projects.

Troubleshooting

Browse frequently asked questions, learn how to resolve some common issues with ActiveReports, and find information on technical support options.

Introducing ActiveReports 6

ActiveReports leverages the latest technologies including Silverlight, XML, scripting and CSS along with open

architecture to provide you with a fully integrated user-friendly report designer. This version now supports Visual Studio 2010, as well as Visual Studio 2005 and 2008.

This section contains information about:

What's New

Learn about the new features in ActiveReports 6.

ActiveReports Editions

Find out which features are freed from the evaluation banner with the Standard and Professional Edition licenses.

GrapeCity Copyright Notice

Explains GrapeCity copyright information.

ActiveReports License Agreement

Understand the terms of the ActiveReports License Agreement and Limited Warranty.

What's New

ActiveReports 6 contains many new features that enhance the reporting capabilities already praised by developers in previous versions of ActiveReports.



You can view the release history notes [here](#). To open the release history notes, click **Show Details** against the ActiveReports 6 version.

What's new in the ActiveReports 6 Service Pack 3?

Underscore Support

You can use the underscore character in the **OutputFormat** property to add the white space after the character specified in the **Value** property. For example, using the underscore character adds the white space for a closing parenthesis in a positive number format while the negative number format includes parentheses. This way both positive and negative values are lined up at the decimal point.

Learn More | OutputFormat Property (on-line documentation)

TextBox Date Values Export Support in Excel

A date value of the **TextBox** control is exported to Excel as a date if the **Value** property is set to a supported date type and the **OutputFormat** property is set to a valid value.

Learn More | OutputFormat Property (on-line documentation)

New Encoding Property

The new **Encoding** property of QRCode allows you to change the text encoding that QRCode uses. The default value is `Encoding.GetEncoding(932)`. You can set this property at design time or in code.

Learn More | Encoding Property (on-line documentation)

New LabelDistanceFactor Property

The new **LabelDistanceFactor** property of the `DataDynamics.ActiveReports.Chart.Marker` class allows you to specify the distance of the Marker label from the chart. The default value is 1, which defines the original distance. If you set the value to 5, the distance is calculated by multiplying 5 times the original value.

This property is provided for the Doughnut/Doughnut3D chart type only.

Learn More | Chart Types | LabelDistanceFactor Property (on-line documentation)

New DateTimeUnit Enumeration

The new **DateTimeUnit** enumeration defines a DateTime unit for the DateTime axis. The default value is **Auto** and it calculates the DateTime unit automatically, based on the data source. The other DateTimeUnit values are **Day**, **Hour**, **Millisecond**, **Minute**, **Month**, **Second** and **Year**. You can set the DateTime unit under Axis properties of the Chart Designer or in code in the DateTimeUnit property.

Learn More | [DateTimeUnit Enumeration \(on-line documentation\)](#) | [DateTimeUnit Property \(on-line documentation\)](#)

Shortcuts in Thumbnails

You can use shortcuts in the Thumbnails pane of the Windows Form Viewer sidebar, the Flash Viewer sidebar and the Silverlight Viewer sidebar. The shortcuts include **Up**, **Down**, **Left**, **Right**, **Page Up**, **Page Down**, **Home** and **End** keyboard shortcuts.

Learn More | [Flash Viewer Hot Keys and Shortcuts](#) | [Silverlight Viewer Hot Keys and Shortcuts](#)

ActiveReports Editions

ActiveReports 6 is an enhancement of the popular ActiveReports engine and report viewer. It includes the same power and flexibility of ActiveReports and the same integration with the Visual Studio® .NET 2005 Environment, and adds several new features including integration with the Visual Studio .NET 2008, Visual Studio .NET 2010 Environments.

Available in two editions, ActiveReports 6 delivers outstanding reporting capabilities. Drop down the sections below to see the features packed with each edition of ActiveReports.

Standard Edition Features

Designer

- Full integration with the .NET environment
- Familiar user interface
- **NEW** SnapLines to help you visually align controls
- C# and VB.NET support
- The ability to compile reports into the application for speed and security or to keep them separate for ease of updating
- Designer hosting of .NET and user controls

Report Controls

- ReportInfo
- Label
- Line
- PageBreak
- OleObject
- Subreport
- Shape
- Picture
- RichTextBox with HTML tag support
- ChartControl with separate data source
- Textbox
- Barcode with standard styles plus **NEW** RSS and UPC styles
- Checkbox

- **NEW** CrossSectionBox extends from a header section to the related footer section
- **NEW** CrossSectionLine extends from a header section to the related footer section

Reporting Engine

- Managed code
- Binding to ADO.NET, XML, iList, and custom data sources
- **NEW** Document DLL as a separate assembly
- All of the features of previous versions of ActiveReports

Report Viewer

- Managed C# code
- Very small deployment assembly, suitable for use on the Internet
- Table of Contents and Bookmarks
- Thumbnail View
- HyperLinking
- Tabbed Viewing
- Annotations

Export Filters

ActiveReports includes export filters to generate output into Rich Text Format (RTF) for word-processing, Portable Document Format (PDF) for printing, 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 6 Professional Edition includes all of the features of the Standard Edition and supports the following additional features:

End-User Report Designer

The 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 for saving and loading report layouts, monitoring and controlling the design environment, and customizing the look and feel to the needs of end users.

ASP.NET Integration

- The 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.

Silverlight Viewer Control

- The Silverlight viewer control allows you to provide in- or out-of-browser report viewing in your Silverlight applications.
- Like our other viewers, the Silverlight viewer control offers customization and localization.

Web Viewer Control

- The Web Viewer control allows quick viewing of ActiveReports on the web as well as printing capability with the AcrobatReader ViewerType enumeration.
- **NEW** Flash ViewerType enumeration supports multiple browsers and offers customization and localization options. The FlashViewer control replaces the ActiveX Viewer control that is no longer supported in ActiveReports 6.

HTTP Handlers

- The RPX HTTPHandler allows the developer to hyperlink ActiveReports on a web page to return HTML format or PDF format reports for viewing and/or printing.
- The Compiled Report HTTPHandler allows the developer to hyperlink ActiveReports compiled in an assembly on a web page to return HTML format or PDF format reports for viewing and/or printing.

PdfSignature and TimeStamp Features

- The PdfSignature class allows you to provide PDF document digital signatures and certification.
- The PdfStamp class allows you to draw the digital signatures and certification onto the documents.
- The TimeStamp class allows you to add a TSA (Time Stamping Authority) stamp to your digital signatures.

Font Linking

- Font linking helps you resolve the situation when fonts on a deployment machine do not have the glyphs that were used in a development environment.
- By linking fonts, you can resolve the problem with a different PDF output on deployment and development machines that may occur due to the missing glyphs.

Font Fallback

- If missing glyphs are not found in linked fonts, the PDF export filter looks for the glyphs in fonts declared in the **FontFallback property (on-line documentation)**.
- A default font is used if you do not declare one, or you can declare an empty string for this property to leave out missing glyphs from the exported file.

Bold Font Emulation (PDF Export Filter)

Some fonts (for example, Franklin Gothic Medium, Microsoft Sans Serif, most East Asian fonts, etc.) may lose bold style in the PDF output. By using the **Font Fallback** mechanism, ActiveReports 6 Professional Edition provides the bold style emulation in the PDF export filter to eliminate this limitation.

GrapeCity Copyright Notice

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. No part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photo copying, recording, or otherwise), or for any purpose, without the express written permission of GrapeCity, inc.

The ActiveReports License Agreement constitutes written permission for Professional Edition licensees to copy documentation content for distribution with their end user designer applications so long as GrapeCity is given credit within the distributed documentation.

ActiveReports and the ActiveReports logo are registered trademarks of GrapeCity, inc.

All other trademarks are the property of their respective owners.

ActiveReports License Agreement

GrapeCity License Agreement and Limited Warranty

Please read carefully before installing this software package. Your installation of the package indicates your acceptance of the terms and conditions of this license agreement. Contact GrapeCity, inc. if you have any questions about this license.

The Product (libraries and object code, and all updates and enhancements thereto that may be released at GrapeCity, inc.'s sole discretion) is proprietary to GrapeCity, inc. and is protected by Federal Copyright Law. GrapeCity retains the title to and ownership of the Product.

If the Product you have obtained is marked as a "TRIAL" or "EVALUATION," you may install one copy of the Product(s) for evaluation purposes for a period of days from the date of installation ("Evaluation Period"). Upon expiration of the Evaluation Period, the Product(s) will cease to function. The Trial version of this product must not be used in Licensee's business applications.

You are licensed to use this Product on the following terms and conditions:

1. GRANT OF LICENSE:

1a. **SITE LICENSE.** If you purchased a Site License ("Site License") version of this product: This Product is licensed per "Site". A "Site License" is defined for all developers at a single physical location (physical mailing address). Licensee is defined as the person or entity that pays consideration for the license to use the Product. GrapeCity, inc. hereby grants the Licensee a nonexclusive License authorizing all developers at a single physical location (one mailing address) to use the Product for development purposes. The use of this License does not create any kind of partnership or joint ownership interest in the Product. Licensee may incorporate the sample code into Licensee's applications. Use of this product at more than one physical location (physical mailing address) at a time terminates, without notification, this License and the right to use the Product. Licensee may not network the Product or otherwise use it on more than one physical location (physical mailing address) at the same time unless Licensee has an appropriate license. Please contact GrapeCity, inc. if you require additional Site Licenses.

1b. **DEVELOPER LICENSE.** If you purchased a "Developer License" of this product: The Product is licensed per software application developer ("developer"). Licensee is defined as the person or entity that pays consideration for the license to use the Product. GrapeCity, inc. hereby grants the Licensee a nonexclusive License authorizing one, and only one, developer at a time to use the Product for development purposes. The use of this License does not create any kind of partnership or joint ownership interest in the Product. Licensee may incorporate the sample code into Licensee's applications. Use of this product by more than one developer at a time terminates, without notification, this License and the right to use the Product. Licensee may not network the Product or otherwise use it on more than one computer terminal at the same time unless Licensee has an appropriate server license. Please contact GrapeCity, inc. if you require additional Licenses.

1c. **SOURCE CODE LICENSE.** If you purchased the source code ("Source Code") license of this product: The "Source Code" is licensed per developer. The "Source Code" can reside on that developer's desktop/work computer and/or that developer's notebook computer. The "Source Code" may not reside in a developer's shared folder(s) or on a network storage device.

ADDITIONAL RESTRICTIONS RELATING TO SOURCE CODE – In addition to the restrictions set forth above, Licensee may use the Product source code ("Source Code") subject to the following limitations:

1. The Source Code is provided to Licensee for the sole purposes of education and troubleshooting related to the use and integration of the Product. The Source Code cannot be used as a basis of development of derivative works or other software products or applications.
2. Under no circumstances may any portion of the Source Code or any modified version of the Source Code be distributed, disclosed or otherwise made available to any third party.
3. GrapeCity, inc. will not provide any support for the Source Code, beyond the initial setup and compilation process.

4. Licensee hereby grant to GrapeCity, inc., a worldwide, irrevocable, perpetual, fully paid-up, unlimited license under any and all intellectual property rights Licensee may have in any modifications to the Source Code made by Licensee, to use, disclose, modify, reproduce, license, sublicense, distribute, commercialize and to otherwise freely exploit such modifications, directly or indirectly, without restriction of any kind.
 5. Licensee will take necessary and appropriate action by instruction, agreement, or otherwise with those of its employees, agents, contractors, or sub-contractors having access to the Source Code to restrict and control the use, copying, modification, protection and security of the Source Code in accordance with this Agreement, including, but not limited to, limiting access to the Source Code to those of Licensee's employees, agents, contractors, or sub-contractors who either have a need to know or who are directly engaged in the maintenance or enhancement of the Product.
2. LICENSEE MAY NOT: Distribute, rent, sub-license or otherwise make available to others the software or documentation or copies thereof, except as expressly permitted in this License without prior written consent from GrapeCity, inc. In the case of an authorized transfer, the transferee must agree to be bound by the terms and conditions of this License Agreement.
3. RESTRICTIONS: Licensee may use the Product in Licensee's business application for sale or distribution as long as: (a) The application that Licensee produces and/or distributes is NOT a software development application that is sold primarily to software developers or system integrators or a development environment of any kind. Please contact GrapeCity, inc. for special commercial licensing provisions in these circumstances; (b) The software serial number and Licensee is registered with GrapeCity, inc.; (c) Licensee does not remove any proprietary notices, labels, or trademarks on the Product or documentation; (d) Licensee does not copy documentation content for distribution with Licensee's application unless GrapeCity, inc. is given credit within the distributed documentation; and (e) Licensee does not modify, de-compile, disassemble, reverse engineer or translate the Product or any component thereof.
4. FILES THAT MAY BE DISTRIBUTED WITH LICENSEE APPLICATION: Licensee may not distribute any files contained in the Product other than the following, each of which may be distributed, on a royalty-free basis, with Licensee's application subject to the restriction set forth above: ActiveReports6.dll, ActiveReports.Document.dll, ActiveReports.Chart.dll, ActiveReports.Viewer6.dll, ActiveReports.Interop.dll, ActiveReports.Interop64.dll, ActiveReports.XlsExport.dll, ActiveReports.HtmlExport.dll, ActiveReports.PdfExport.dll, ActiveReports.RtfExport.dll, ActiveReports.TextExport.dll, ActiveReports.TiffExport.dll.
- Professional Edition: ActiveReports.Design6.dll, ActiveReports.FlashViewer.swf, ActiveReports.FlashViewer.Resources.swf, any *.swf file from the Themes folder, ActiveReports.Web.dll, and ActiveReports.Silverlight.dll, in addition to the Standard Edition files.
- Also, [AR6DesignerGuide.pdf](#), AR6DesignerGuide.chm, or any output from the ActiveReports6EUD.hsp file available as a [zipped file](#) on the Web site may be distributed as documentation for your licensed End User Designer application so long as GrapeCity is given credit within the distributed documentation.
5. 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.
6. TERMINATION: Licensee may terminate the License at any time by destroying all copies of the Product and Product documentation. This License will also terminate automatically if Licensee fails to comply with any term or condition in this Agreement.
7. LIMITED WARRANTY: Licensee assumes all responsibility for the selection of the Product as appropriate to achieve the results Licensee intends. GrapeCity, inc. warrants that the enclosed diskette or compact disc medium upon which the Product is recorded shall be free from defects in material and workmanship under normal use and conditions, and that the Product shall perform substantially as described in its documentation for a period of ninety (90) days from purchase. This Limited Warranty is void if failure of the Product has resulted from accident, abuse, or misapplication. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, THE PRODUCT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
8. CUSTOMER REMEDIES. GrapeCity, inc.'s entire liability and Licensee's exclusive remedy shall be, at GrapeCity, inc.'s option, either (a) return of the price paid or (b) repair or replacement of the Product that does not meet GrapeCity, inc.'s Limited Warranty and which is returned to GrapeCity, inc. with a copy of your receipt. Any replacement Product will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

9. **NO LIABILITY FOR CONSEQUENTIAL DAMAGES.** In no event shall GrapeCity, inc. or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of use of or inability to use the Product, even if GrapeCity, inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of certain implied warranties or the limitation of the foregoing warranty or the exclusion or limitation of incidental or consequential damages, in which case and to the extent such exclusion or limitation is not allowed some of the foregoing limitations and exclusions may not apply to you.

10. **GOVERNING LAW; VENUE:** This Agreement shall be governed by, and construed in accordance with, the laws of the State of Washington and the United States, and any action brought in connection with this Agreement shall be brought only in the state or federal courts sitting in Seattle, Washington. In any such action, you submit to the personal jurisdiction of such courts and waive any objections to venue in such courts. If any action is brought by either party to this Agreement against the other party regarding the subject matter hereof, the prevailing party shall be entitled to recover, in addition to any other relief granted, reasonable and actual attorney fees and expenses of litigation. Application of the U.N. Convention of Contracts for the International Sale of Goods is expressly excluded. The original version of this Agreement is the English language version. Any discrepancy or conflicts between the English version and the versions in any other language will be resolved with reference to and interpreting the English version, which will control.

Installation

This section will help you to understand the installation process.

This section contains information about:

Requirements

Learn about the hardware and software required to run ActiveReports 6.

ActiveReports and the .NET Framework Compatibility

Learn about the compatibility of the ActiveReports 6 assemblies with the .NET Framework versions.

Installed Files

Find out what files are installed with ActiveReports 6, and where to locate them.

Installation Troubleshooting

Get help with installation issues.

Service Packs and Hot Fixes

Learn about our interim product releases.

ActiveReports for .NET 2.0 Side-by-Side Installation

Learn about working with ActiveReports 6 and ActiveReports for .NET 2.0 on a single machine.

Installing ActiveReports 6 Help into Visual Studio 2010

Find out how to integrate ActiveReports 6 Help into Visual Studio 2010.

Installer Command-Line Options

Learn about the command line options you may use with the installer.



Tip: At the end of the installation process, you can open the Readme.hta file by leaving the **Review release notes** check box selected. After installation, you can locate it in the **Introduction** folder of the root ActiveReports 6 installation folder. Double-click to open and view the file.

Requirements


To install and use ActiveReports 6, you need compatible hardware and software.


Hardware requirements (minimum)

- **Processor:** Pentium® II-class processor 450 MHz (Pentium® III 600 MHz recommended)
- **RAM:** 200 MB
- **Hard drive space:** 50 MB available

Software requirements

- **Operating System:** Windows® 2000, Windows® XP, Windows® NT 4.0, Windows™ Vista, Windows 7, Windows Server 2003, Windows Server 2008, or Windows Server 2008 R2.
- **Microsoft® .NET Framework Version:** 2.0 or higher.
- **Microsoft Visual Studio:** 2005, 2008, or 2010.

 **Note:** The Express Editions of Visual Studio do **not** work with ActiveReports, as they do not support packages.

 **Note:** Microsoft Silverlight 4 Tools is required for the proper application development with the ActiveReports Silverlight Viewer.

- **For Web deployment:** IIS 5.0, 5.1, 6.0, 7.0 or 7.5 and ASP.NET (version to match the .NET Framework version).

ActiveReports and the .NET Framework Compatibility

All ActiveReports 6 assemblies are compliant with .NET Framework 3.5 Full profile and .NET Framework 4.0 Full profile.

The following assemblies are compliant with .NET Framework 3.5 Client Profile and .NET Framework 4.0 Client Profile:

| File | Description |
|------------------------------|---|
| ActiveReports6.dll | Run-time engine assembly file. |
| ActiveReports.Chart.dll | Chart control assembly file. |
| ActiveReports.Document.dll | Document assembly file. |
| ActiveReports.Interop.dll | Native functions assembly file. |
| ActiveReports.PdfExport.dll | PDF Export assembly file. |
| ActiveReports.RtfExport.dll | RTF Export assembly file. |
| ActiveReports.TextExport.dll | Text Export assembly file. |
| ActiveReports.TiffExport.dll | TIFF Export assembly file. |
| ActiveReports.Viewer6.dll | Viewer assembly file. |
| ActiveReports.XlsExport.dll | Microsoft® Excel® Export assembly file. |

The following assemblies are not compliant with .NET Framework 3.5 Client Profile and .NET Framework 4.0 Client Profile:

| File | Description |
|------------------------------|----------------------------|
| ActiveReports.Design6.dll | Designer assembly file. |
| ActiveReports.HtmlExport.dll | HTML Export assembly file. |
| ActiveReports.Web.dll | Web assembly file. |

The **End User Report Designer**, the **WebViewer** control and the **HTML Export** filter require the full profile.

Installed Files

To verify package installation

1. Open Visual Studio®.
2. If the package installed successfully, the ActiveReports logo is on the splash screen.
3. From the **Help** menu, select **About** and verify that **ActiveReports** appears in the installed products list.

When you install ActiveReports and use all of the default settings, files are installed in the following folders:

C:\Documents and Settings\YourAccountName\Start Menu\Programs\GrapeCity\ActiveReports 6

File (or Folder)

Samples

ActiveReports 6 Documentation for Visual Studio .NET 2005

ActiveReports 6 Documentation for Visual Studio .NET 2008

ActiveReports 6 Documentation for Visual Studio .NET 2010

Introduction

Uninstall ActiveReports 6

Description

Start menu shortcuts to included sample projects.

Shortcut to the integrated help file.

Shortcut to the integrated help file.

Shortcut to the integrated help file.

Shortcut to the readme.hta file.

Shortcut to the installer application.

C:\Documents and Settings\YourAccountName\My Documents\GrapeCity\ActiveReports 6

File (or Folder)

Samples

Description

Included sample projects.

C:\Program Files\GrapeCity\ActiveReports 6 (C:\Program Files (x86)\GrapeCity\ActiveReports 6 on a 64-bit Windows operating system)

File (or Folder)

Description

Data

Sample XML and MDB data files.

Deployment

Flash viewer file, Flash viewer themes, Silverlight localization resources and templates for redistribution.

Introduction

Readme.hta and associated image files.

Localization

Resource and DOS batch files for localizing ActiveReports components. For more information, see **Localize Active Reports Resources**.

C:\Program Files\Common Files\GrapeCity\ActiveReports 6 (C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6 on a 64-bit Windows operating system)

File (or Folder)

1033

Description

Folder for default U.S. English locale.

1041

Folder for Japanese locale.

ActiveReports6.dll

Run-time engine assembly file.

ActiveReports.Chart.dll

Chart control assembly file.

ActiveReports.CodeDomSerializer.dll

Helper file for Visual Studio integration.

ActiveReports.Design6.dll

Designer assembly file.

ActiveReports.Document.dll

Document assembly file.

ActiveReports.HtmlExport.dll

HTML Export assembly file.

| | |
|--|---|
| ActiveReports.Interop.dll | Native functions assembly file. |
| ActiveReports.PdfExport.dll | PDF Export assembly file. |
| ActiveReports.RtfExport.dll | RTF Export assembly file. |
| ActiveReports.Silverlight.Design.dll | Silverlight designer assembly file. |
| ActiveReports.Silverlight.dll | Silverlight assembly file. |
| ActiveReports.TextExport.dll | Text Export assembly file. |
| ActiveReports.TiffExport.dll | TIFF Export assembly file. |
| ActiveReports.Viewer6.dll | Viewer assembly file. |
| ActiveReports.Web.Design.dll | Web designer assembly file. |
| ActiveReports.Web.dll | Web assembly file. |
| ActiveReports.XlsExport.dll | Microsoft® Excel® Export assembly file. |
| AR6Col_A.HxK | ActiveReports help integration index. |
| ARVSPackage.dll | Visual Studio® integration package. |
| COL_*. * (14 files) | ActiveReports help integration collection. |
| ddAR6.HxS | ActiveReports 6 Help File. |
| GrapeCity.ActiveReports.Silverlight.License.KeyGen.AddIn.AddIn | Silverlight viewer runtime license key generator add-in file. |
| GrapeCity.ActiveReports.Silverlight.License.KeyGen.AddIn.dll | Silverlight viewer runtime license key generator assembly file. |
| H2Reg*. * (3 files) | Files used to register the ActiveReports help collection with the combined help collection of Visual Studio. |
| riched20.dll | Version 4.0 of the Microsoft® RichEdit control, used to support rtf tables in edit mode of the ActiveReports RichTextBox control. |

C:\Program Files\Common Files\GrapeCity\ActiveReports 6\Redist (C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6\Redist on a 64-bit Windows operating system)

| File (or Folder) | Description |
|-----------------------------|--|
| ActiveReports.Interop64.dll | Native functions assembly for 64-bit machines. |

Installation Troubleshooting

Symptoms: Other users cannot access or use ActiveReports on my machine.

Cause: The installation for ActiveReports 6 gives you the option to install the program for everyone or the current user. If it is installed only for the current user, other users cannot access it or use it.

Solution: Reinstall ActiveReports and select Everyone.

Symptoms: I just installed ActiveReports 6. Why can't I see the help files?

Cause: If the installation was run while Visual Studio® was open, the help files cannot be integrated.

Solution: Close Visual Studio and reopen it.

Symptoms: When I run the ActiveReports Setup, I get the message "The installer was interrupted before GrapeCity ActiveReports 6 could be installed. You need to restart the installer to try again."

Cause: You do not have permissions to install on the machine, or to the folder containing the setup files.

Solution: Verify that the system account for the local machine has permissions to the folder containing the setup and log in as Administrator on the machine.

Service Packs and Hot Fixes

We are always improving our products, and from time to time we release Service Packs or Hot Fixes on our Web site. These are always free of charge.

Hot Fixes

These are interim releases that have fixes for specific issues found either internally or reported by users. Hot Fixes are tested, but not as rigorously as Service Packs. We recommend that you install a Hot Fix only if you are affected by an issue that is fixed in it and cannot wait for a Service Pack.

Service Packs

These are interim releases that include all fixes incorporated in the Hot Fixes up to that point in time, plus we add a few minor features to each one. Service Packs undergo the same rigorous testing as product releases.

To install a Hot Fix or Service Pack

1. Close Visual Studio and any Visual Studio help files you may have open.
2. Download and run the **ActiveReports6.exe** file.
3. The installer detects the previous version, and asks permission to uninstall it. Click **OK**.
4. When it finishes uninstalling, select the options to install the new build as usual. (The installer automatically brings forward your licensing information, so you are not prompted for it again.)
5. Open Visual Studio, and in the toolbox, right-click and select **Choose Items**.
6. In the Choose Toolbox Items dialog that appears, in the Filter box, enter **DataDynamics**. The list displays only DataDynamics components.
7. Drag the right edge of the **Assembly Name** column so that you can see the build number.
8. Clear the checkbox next to any component using an old assembly, and select the checkbox for the component in the new assembly.
9. Click **OK**. The new versions of the components are added to the toolbox.

To update a WebViewer project after installing a Hot Fix or Service Pack

1. In the **Web.config** file, update the version number in the following HTTP Handler code.

```
***** ActiveReports HttpHandler Configuration *****
-->
    <add verb="*" path="*.rpx" type="DataDynamics.ActiveReports.Web.Handlers.Rp
xHandler, ActiveReports.Web, Version=6.0.1661.0, Culture=neutral, PublicKeyToken=cc
4967777c49a3ff"/>
    <add verb="*" path="*.ActiveReport" type="DataDynamics.ActiveReports.Web.Ha
ndlers.CompiledReportHandler, ActiveReports.Web, Version=6.0.1661.0, Culture=neutra
l, PublicKeyToken=cc4967777c49a3ff"/>
    <add verb="*" path="*.ArCacheItem" type="DataDynamics.ActiveReports.Web.Han
```



```
dlers.WebCacheAccessHandler, ActiveReports.Web, Version=6.0.1661.0, Culture=neutral
, PublicKeyToken=cc4967777c49a3ff"/>
</httpHandlers>
```

2. Open the ASPX page and look in the **Source** view for a line that looks similar to the following and update the version number:

```
<%@ Register TagPrefix="ActiveReportsWeb" Namespace="DataDynamics.ActiveReports.Web"
"
    Assembly="ActiveReports.Web, Version=6.0.1661.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" %>
```

3. Save and rebuild your project.

ActiveReports for .NET 2.0 Side-by-Side Installation

ActiveReports 6 does not support a side-by-side installation with ActiveReports for .NET 2.0, therefore we do not recommend to install both products on a single machine.

If you decide to have both products installed on a single machine, you will encounter the following compatibility issues:

1. The **Add New Item** dialog may create the ActiveReports 6 xml-based report template instead of the ActiveReports for .NET 2.0 report template after ActiveReports 6 has been installed.
2. ActiveReports 6 xml-based reports may be opened in the Active Reports for .NET 2.0 report designer after ActiveReports for .NET 2.0 has been installed on top of ActiveReports 6.

Switching between ActiveReports for .NET 2.0 and ActiveReports 6

ActiveReports 6 includes the Rpx.Designer.Switcher tool that allows you to run both ActiveReports for .NET 2.0 and ActiveReports6 on the same machine.

To run the Rpx.Designer.Switcher tool:

1. Close all instances of Visual Studio .NET 2005.
2. On the **Start** menu, choose **Run**.
3. Click the **Browse** button to select the Rpx.Designer.Switcher tool (by default, the Rpx.Designer.Switcher tool is located at C:\Program Files\Common Files\GrapeCity\ActiveReports 6; on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6).
4. Execute the Rpx.Designer.Switcher.exe by clicking **OK** in the **Run** dialog box.
5. Finally, choose the needed product in the **ActiveReports6 Switcher Utility** dialog box and click **OK**.

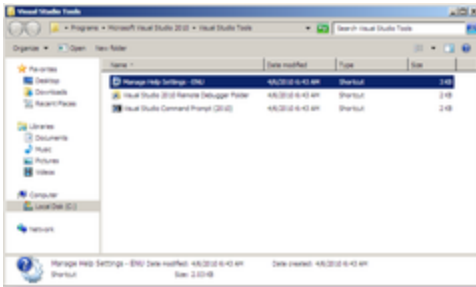


Note: Do not run the Rpx.Designer.Switcher tool while Visual Studio .NET 2005 is running.

Installing ActiveReports 6 Help into Visual Studio 2010

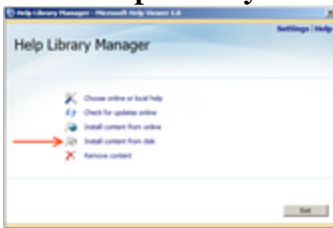
You can integrate ActiveReports 6 Help into Visual Studio 2010 on your computer and have access to ActiveReports User Guide any time you work in Visual Studio 2010.

1. To launch the **Help Library Manager**, open **Manage Help Settings** in the **Microsoft Visual Studio 2010/Visual Studio Tools** folder.

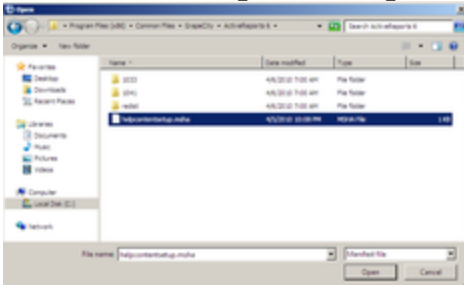


Note: You can also launch the **Help Library Manager** by selecting **Manage Help settings** in the **Help** menu of Visual Studio 2010.

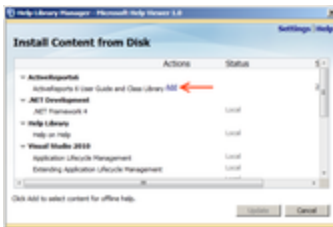
2. On the **Help Library Manager** page, click **Install content from disk**.



3. Navigate to the **ActiveReports 6 Help** manifest file (by default, the file is located at C:\Program Files\Common Files\GrapeCity\ActiveReports 6; on a **64-bit Windows operating system** the default location is C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6).
4. Select the **ActiveReports 6 Help** manifest file and click **Open**.



5. Click **Next** on the displayed **Install Content from Disk** page.
6. Click the **Add** action next to ActiveReports 6 User Guide and Class Library and then click the **Update** button.



7. In the displayed **Security Alert** dialog, click **Yes**.
8. Finally, click **Finish** on the displayed **Updating Local Library** page to finish the ActiveReports 6 Help installation in Visual Studio 2010.

Note: If you still receive the 404 error message when you try to open the ActiveReports 6 Help, we recommend that you restart the Help Library Agent or simply close it in the notification area.

For more information on Help Troubleshooting, please see **Troubleshooting**.

Installer Command-Line Options

You can pass options to ActiveReports6 Setup.exe via the command line.

1. Go to the **Start** menu.
2. Click **Run**, enter cmd in the Run dialog and click OK.
3. In the Command window that appears, enter the following command where you may use options from the table below.

Enter into the Command window


```
"C:\ActiveReports6\ActiveReports 6 Setup.exe" /s [OPTION1=VALUE1] [OPTIONN=VALUEN]
```

| Option | Type | Description |
|-----------------------|---------|--|
| TARGETDIR | string | The installation path for ActiveReports 6. |
| MINIMUM | boolean | Compact installation with the minimum required features (the samples and documentation are not included). Note: Set to TRUE or FALSE in uppercase, otherwise the value will not take effect. |
| COMPLETE | boolean | Complete installation (the samples and documentation are included). Note: Set to TRUE or FALSE in uppercase, otherwise the value will not take effect. |
| INSTALLSAMPLES | boolean | Specifies whether to install the samples. Note: Set to TRUE or FALSE in uppercase, otherwise the value will not take effect. |
| SAMPLESDIR | string | The installation path for the samples. If TARGETDIR is not set, the default directory is "MYDOCUMENTS\GrapeCity\ActiveReports 6\Samples". If TARGETDIR is set, the default directory is the one you specify. |
| ALLUSERS | boolean | Performs installation for all users if set to TRUE . Otherwise, if set to FALSE , performs installation for the current user only. |
| USERNAME | string | The user name information for customer registration. |
| EMAIL | string | The email information for customer registration. |
| SERIAL | string | If SERIAL is not set, an evaluation version will be installed. |
| PROXYSERVER | string | The proxy server information. |
| PROXYPORT | string | The proxy port information. |
| PROXYUSERNAME | string | The proxy username information. |
| PROXYPASSWORD | string | The proxy password information. |

License Your ActiveReports

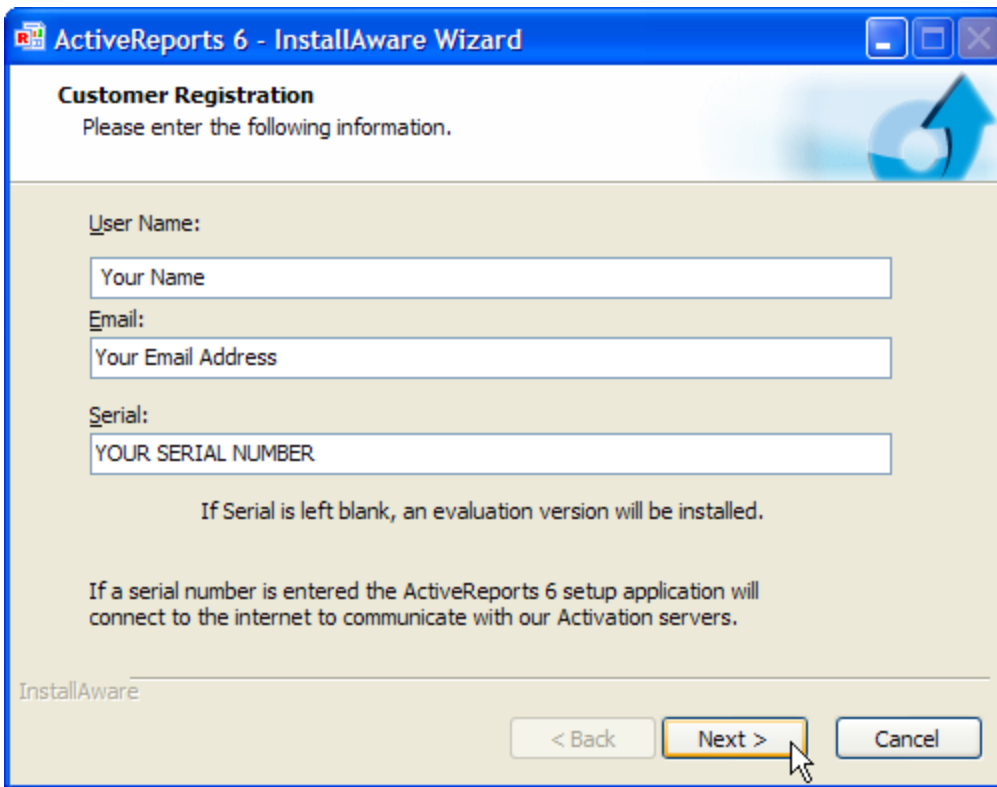
License your ActiveReports during installation, or from Add or Remove Programs if you already have a trial version installed.

When you install a service pack or hot fix on top of a previous version of ActiveReports 6, the Customer Registration screen is not shown during installation. If your previous version was not licensed and you wish to license it now, follow the steps described in **To license a trial version of ActiveReports without reinstalling** below.

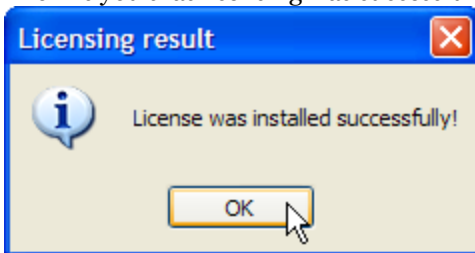
 **Caution:** When installing on a Vista machine, do not just double-click the exe to install ActiveReports. This will not license the machine. On Windows Vista, you must instead right-click the exe and select **Run as Administrator** even if you are already the administrator on your machine.

To license a machine for ActiveReports during installation

1. Near the end of the installation, the Customer Registration screen requests the following information:
 - **User Name:** Enter your name or company name here. You can use any characters in this field except the semicolon.
 - **Email:** Enter your e-mail address in this field.
 - **Serial:** Enter the serial number exactly as you received it from GrapeCity, including any dashes or capital letters.



2. Click the **Next** button and then the **Finish** button to complete the installation. The Licensing result message box informs you that licensing was successful.



3. The machine is now licensed, and no nag screens or evaluation banners appear when you use the product or create new solutions with it.

To license a trial version of ActiveReports without reinstalling

1. From the Start menu, open the **Control Panel**.
2. Select **Add or Remove Programs**.
3. From the list of currently installed programs, select **ActiveReports 6**.
4. Click the **Change** button.
5. Select **License ActiveReports 6** and click the **Next** button.



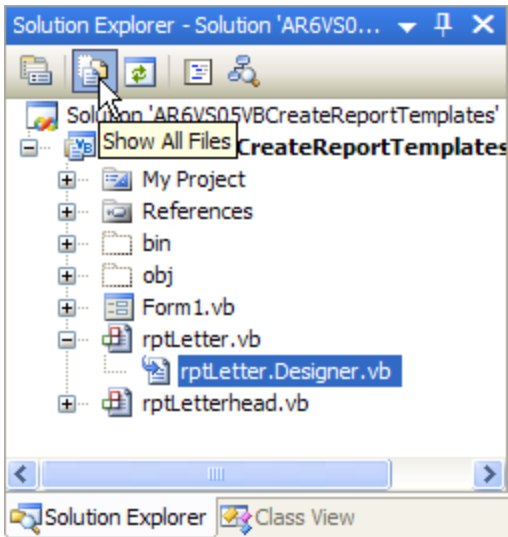
6. The Customer Registration screen requests the following information:
 - **User Name:** Enter your name or company name here. You can use any characters in this field except the semicolon.
 - **Email:** Enter your e-mail address in this field.
 - **Serial:** Enter the serial number exactly as you received it from GrapeCity, including any dashes or capital letters.

If you have already created any Visual Studio projects using ActiveReports components, see the appropriate section on licensing Windows or Web applications below.

To check an existing ActiveReports Windows application for licensing

Caution: If the application containing ActiveReports is not an executable, licensing must be embedded in the calling application, or root level executable, to take effect.

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



- Expand the My Project node. If there is a file called **licenses.licx** in the file list, the ActiveReports application is licensed.
- If the **licenses.licx** file does not appear in your file list, follow the instructions under **To license Windows Forms projects** below.

To license Windows Forms projects made with the trial version

- Ensure that ActiveReports is licensed on the machine by following the steps above for licensing either during installation of ActiveReports or later if using a trial version.
- Open the project in Microsoft Visual Studio.
- Open the Visual Studio **Build** menu and select **Rebuild Solution**.
- The executable application is now licensed, and no nag screens or evaluation banners appear when you run it. You can distribute the application to un-licensed machines and no nag screens or evaluation banners appear.

To license Web Forms projects made with the trial version

- Ensure that ActiveReports is licensed on the machine by following the steps above for licensing either during installation of ActiveReports or later if using a trial version.
- Open the project in Microsoft Visual Studio.
- Open the Visual Studio **Build** menu and select **Rebuild Solution**.

Note: For licensing Web Site applications, open the Visual Studio **Build** menu and select **Build Runtime Licenses** to create the App_Licenses.dll file.

- The web application is now licensed, no evaluation banners appear when you run it. You can distribute the Web application to unlicensed machines and no evaluation banners appear.

Required references in the licenses.licx file (for Standard and Professional Editions)

The licenses.licx file must contain the following references to the ActiveReports 6 version and the reference to the Viewer control:

Standard Edition:

Paste INSIDE the licenses.licx file

```
DataDynamics.ActiveReports.ActiveReport, ActiveReports6, Version=x.x.xxxx.x,  
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

```
DataDynamics.ActiveReports.Viewer.Viewer, ActiveReports.Viewer6, Version=x.x.xxxx.x,  
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

Professional Edition:**Paste INSIDE the licenses.licx file**


```
DataDynamics.ActiveReports.ActiveReport, ActiveReports6, Version=x.x.xxxx.x,
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

```
DataDynamics.ActiveReports.Web.WebViewer, ActiveReports.Web, Version=x.x.xxxx.x,
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

```
DataDynamics.ActiveReports.Export.Pdf.PdfExport, ActiveReports.PdfExport,
Version=x.x.xxxx.x, Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

```
DataDynamics.ActiveReports.Design.Designer, ActiveReports.Design6, Version=x.x.xxxx.x,
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```

```
DataDynamics.ActiveReports.Viewer.Viewer, ActiveReports.Viewer6, Version=x.x.xxxx.x,
Culture=neutral, PublicKeyToken=cc496777c49a3ff
```


 **Note:** The version number (Version=x.x.xxxx.x) must correspond to the Active Reports 6 version you are using.

To create the Web Key with the Web Key Generator utility

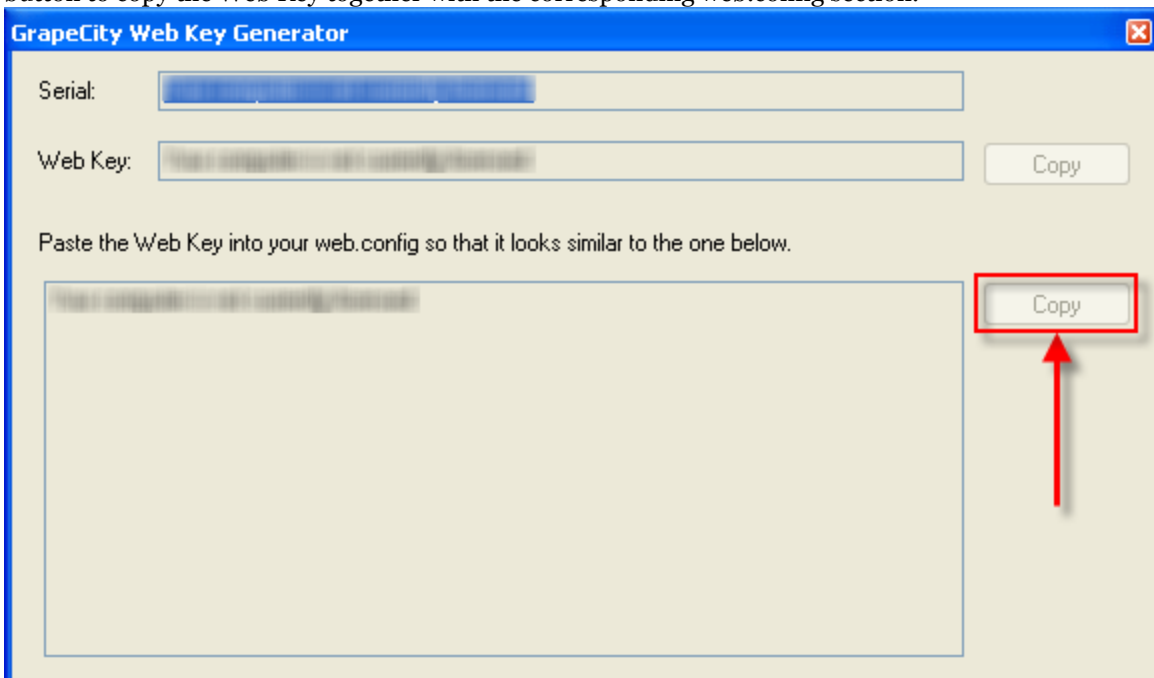
You can create a Web Key for your ActiveReports project by using the Web Key Generator utility on a machine with the licensed ActiveReports.

1. From the Start Menu, go to **All Programs > GrapeCity > ActiveReports 6.0 > Generate WebKey** and run the **Web Key Generator** utility.



 **Note:** You can find the Web Key Generator utility in the C:\Program Files\Common files\GrapeCity\ActiveReports 6 folder.


2. In the dialog that appears, copy the Web Key by clicking **Copy**. We recommend that you use the second **Copy** button to copy the Web Key together with the corresponding web.config section.



- Paste the Web Key into the web.config file of your project between the opening and closing <configuration> tags to remove the licensing message. The web.config key looks like the following.

XML code. Paste INSIDE the Web.config file

```
<configuration>
  <appSettings>
    <add key=" ActiveReportsLicense " value="Name,Company,SERIAL NUMBER,Generated
Hash Code" />
  </appSettings>
</configuration>
```

 **Note:** If you see the message "Your computer is not currently licensed" in the **Web Key Generator** dialog, please license your machine. To do that, follow the steps described in the section **To license a trial version of ActiveReports without reinstalling** that you will find above.

To license an ActiveReport6 Silverlight project

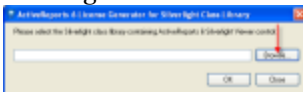
- Ensure that ActiveReports is licensed on the machine by following the steps above for licensing either during installation of ActiveReports or later if using a trial version.
- Open your project in Microsoft Visual Studio.
- On the Visual Studio **Tools** menu, select **Generate ActiveReports6 Silverlight Runtime License**.
- In the dialog that appears, click **OK**.



To license a Silverlight Class Library project with the ActiveReports 6 Silverlight Viewer


You can license your Silverlight Class Library project that contains the ActiveReports 6 Silverlight Viewer control by running the ActiveReport6 Silverlight Class Library License Generator utility from the Start Menu.

- Ensure that ActiveReports is licensed on the machine by following the steps above for licensing either during installation of ActiveReports or later if using a trial version.
- From the Start Menu, go to **All Programs > GrapeCity > ActiveReports 6 > Tools > Silverlight Class Library License Generator** and run the **Silverlight Class Library License Generator** utility.
- In the dialog that appears, click **Browse...** to select the Silverlight class library with the ActiveReports 6 Silverlight Viewer control and click **Open**.





- In the same dialog, click **OK** after selecting the Silverlight class library with the ActiveReports 6 Silverlight Viewer control.
- In the following dialog, click **Save** to save the generated license data to the GrapeCity.ActiveReports.Silverlight.License.lcx license file.



 **Note:** If your Silverlight application project already contains the license file, you can update it with the license data for the Silverlight class library. To do that, use the **Copy** button on the **ActiveReports 6 License Generator for Silverlight Class Library** dialog to copy and then paste the <Product> node from the generated license data to the existing license file.

6. In the dialog that appears, specify the location to save the license file and click **OK**.
7. On the Visual Studio **Project**, click **Add Existing Item** to add the saved license file to the Silverlight class library project.
8. In the dialog that appears, locate and select the GrapeCity.ActiveReports.Silverlight.License.lcx file and click **OK**.
9. In the Solution Explorer, select the GrapeCity.ActiveReports.Silverlight.License.lcx file.
10. In the Properties window, set the **Build Action** property for this license file to **Embedded Resource**.

 **Note:** When using the PDF export filter in your project, you should open the licenses.licx file and make sure that it contains a proper reference to the PDF Export Assembly.

 **Important:** The SetLicense() Method has been deprecated and can no longer be used for licensing ActiveReports. To learn how to license your ActiveReports, please refer to the sections located above.

To license ActiveReports on the machine without internet connection, please contact our support team: powersupport@grapecity.com.

Upgrading Reports

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

This section contains information about:

Changes from Previous Versions

Find out about changes in ActiveReports 6 that may break your old ActiveReports for .NET 1.0, 2.0, or 3.0 projects.

Upgrading Reports from Previous Versions of ActiveReports

Find out how to upgrade reports created with ActiveReports for .NET 1.0, 2.0, or 3.0 to ActiveReports 6.

Migrating from ActiveReports 2

Find out how to upgrade reports created with ActiveReports 2.0 (COM version) to ActiveReports 6.

Converting Microsoft® Access® Reports

Find out about converting Microsoft® Access® reports to ActiveReports.

Converting Crystal Reports

Find out about converting your reports created with 2005, 2008 and 2010 version Crystal Reports to ActiveReports 6.

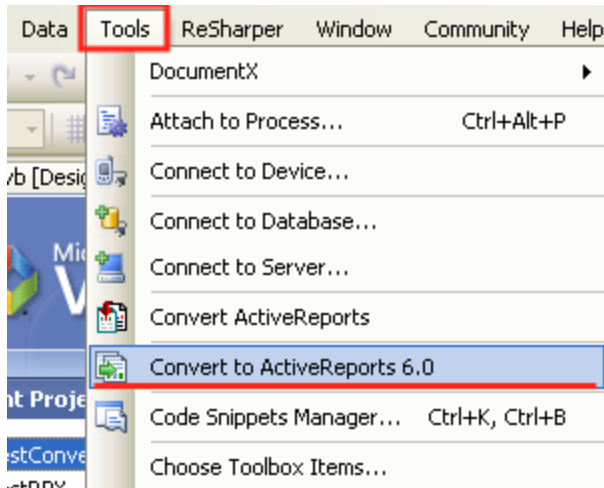
Changes from Previous Versions

There are a few breaking changes introduced in ActiveReports 6 to keep in mind when upgrading your existing ActiveReports for .NET 3.0 or 2.0 projects:

The BaseClass name has been changed.

- **ARN3:** DataDynamics.ActiveReports.**ActiveReport3**
- **AR6:** DataDynamics.ActiveReports.**ActiveReport**

The ActiveReports 6 **Report Converter** converts and updates previous versions of ActiveReports to ActiveReports 6 format. Save back-ups of your reports before running it.



Note: We recommend that you check the Active Reports for NET 2.0 or Active Reports for NET 3.0 project before running the ActiveReports 6.0 **Report Converter** and make sure that the project has valid ActiveReports references.

The **SetLicense** method for run-time reporting and end user designer licensing has been marked as obsolete and raises a compile error. We have updated our licensing models and we sincerely hope to provide an easy and seamless licensing and deployment experience with this release. See **License Your ActiveReports** for more information.

The **Report.Show** method has been removed. This removes the viewer dependency for a leaner package. Instead, you can use the Preview tab at design time or the Viewer control at run time.

The **ActiveX Viewer** control is no longer supported. Instead, we have added the Flash Viewer and more recently the Silverlight Viewer, thus expanding the options of viewing ActiveReports on the web. See **Flash Viewer Options** and **Silverlight Viewer (Pro Edition)** for more information.

Upgrading from Previous Versions

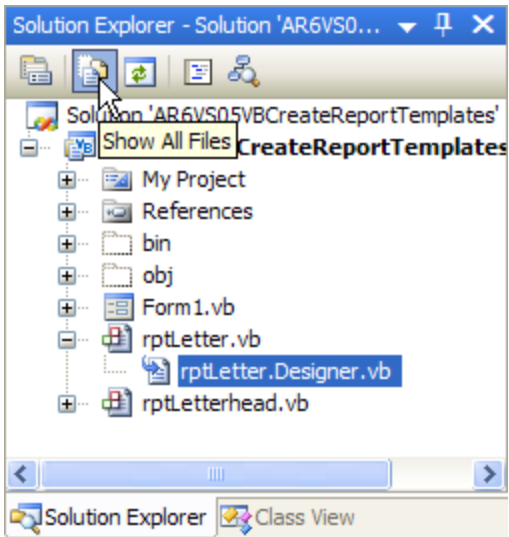
Caution: Always back up your projects before beginning the conversion process.

Upgrading from an ActiveReports for .NET 3.0 project

You can upgrade an ActiveReports for .NET 3.0 project by using the converter utility(see the section **Upgrading from an ActiveReports for .NET 1.0 or 2.0 project** located below), or by opening it in Visual Studio, replacing the old references with the new versions (adding the new Document reference) and correcting two errors in the code.

To update the project references

1. Open the Visual Studio project that contains the reports that you want to upgrade.
2. In the Solution Explorer, click the **Show All Files** button.



3. Expand the **References** folder, and make note of which ActiveReports references you use in your project.
4. Right-click each of the ActiveReports3 references, and select **Remove**.
5. Right-click the References folder and select **Add Reference**.
6. In the Add Reference window that appears, select Version 6.x.xxx.x of the ActiveReports references.

 **Note:** You also need to add the new **GrapeCity ActiveReports Document** reference to the project, as some of the code has moved.

7. Click the **OK** button to add the references and close the window. Many errors appear in the Visual Studio Error List window.


To correct the errors

1. If the Visual Studio Error List window is not showing, drop down the **View** menu and select **Error List**.
2. In the Error List window, double-click the warning that states that **ActiveReports3** could not be defined.
3. In the report code, **ActiveReports3** is highlighted. Change **ActiveReport3** to **ActiveReport**. This resolves most of the errors in the list.
4. In the Error List window, double-click the warning that states that **ActiveReports3.FetchEventArgs** could not be defined.
5. In the report code, change **ActiveReport3** to **ActiveReport**.
6. Close and reopen the design view of the report.

Upgrading from an ActiveReports for .NET 1.0 or 2.0 project

To run the converter utility for old RPX files

Because ActiveReports 6 writes its report layout files in C# or Visual Basic.NET, reports that were built with ActiveReports for .NET 1.0 and 2.0 as RPX files need to be converted to run in the new environment. ActiveReports 6 includes a converter that makes this an easy process.

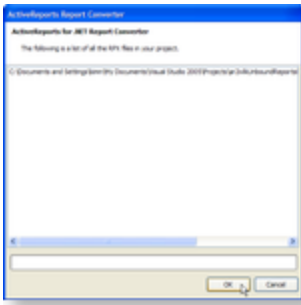
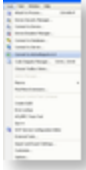
 **Caution:** The converter utility does not support version 1.0 or 2.0 reports that use **inheritance**. To convert these reports, you must first:

- change the **base class** of the inherited report to **ActiveReport**
- ensure that the **InitializeReport** method is free of **Overrides** or similar modifiers

The naming convention for controls changed as of version 3. Previously (in versions 1 and 2) controls could be similarly named by using a different case, as in "Test" and "test," but ActiveReports 6 (and 3) require control names to be unique. When you load an RPX that has two controls with the same names but different cases, the report does not load and an

error occurs. In order to avoid this, give each control a unique name and update the names in the code-behind.

1. Open an existing ActiveReports for .NET 1.0 or 2.0 project in Visual Studio®. The report appears in the Solution Explorer as an RPX file.
2. From the **Tools** menu, select **Convert to ActiveReports 6.0** to open the ActiveReports Report Converter. The converter displays a list of all RPX files in your project, including any in subdirectories.



3. Click **OK** to convert the files. The reports appear in the Solution Explorer as C# or Visual Basic files and all references to earlier versions of ActiveReports are updated.

 **Note:** We recommend that you check the Active Reports for NET 2.0 or Active Reports for NET 3.0 project before running the ActiveReports 6.0 **Report Converter** and make sure that the project has valid ActiveReports references.

4. If the old project used the `rpt.Show` method, an error appears in the Error List window. To correct the error, replace the code with `rpt.Run`, then add a Viewer control to the form and set the `viewer.Document = rpt.Document`. For details, see **Viewing Reports**.

Loading Old RDF Files

The Windows Forms Viewer control can still display RDF files, which are static copies of reports with data from the time at which they were saved, from older versions of ActiveReports. In most cases, the WebViewer can also display many of these files, although certain reports may not display correctly.

Migrating from ActiveReports 2

ActiveReports 6 can use existing ActiveReports 2.0 (COM) 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.

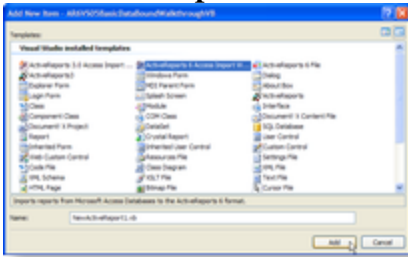
Converting MS Access Reports

You can convert Microsoft® Access® reports into ActiveReports format by running the ActiveReports Microsoft Access Import Wizard. Due to differences between products, the extent to which your reports are converted depends on your specific report layout. However, since GrapeCity provides source code, you can modify the resulting ActiveReport to

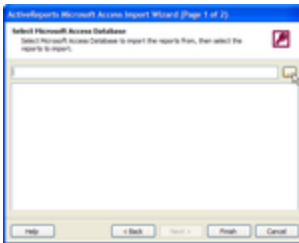
achieve the results you desire.

To convert a Microsoft® Access® report into an ActiveReport

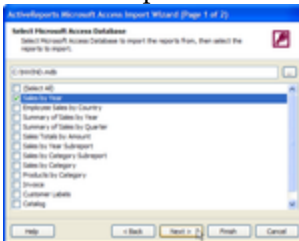
1. Open a project in Visual Studio®.
2. From the **Project** menu, select **Add New Item**.
3. Select **ActiveReports 6 Access Import Wizard** from the Templates pane and click the **Add** button.



4. In the ActiveReports Microsoft Access Import Wizard that appears, click the ellipsis button to browse for the Access Database that contains the report or reports you want to convert and click the **Open** button.



5. If you receive a security warning, click **Open** to proceed.
6. Select the reports from the database that you want to import and click **Next**.



7. Click **Finish** to begin the conversion process.
8. Click **Open** to proceed through any security warnings.
9. Click **Finish** when the conversion process has finished. The converted reports appear in the Solution Explorer.

Converting Crystal Reports

You can convert Crystal Reports reports into ActiveReports format by running the Crystal Reports to ActiveReports Converter Wizard. Due to differences between products, the extent to which your reports are converted depends on your specific report layout.

To run the Crystal Reports to ActiveReports Converter Wizard, you need to install Visual Studio and Crystal Reports for Visual Studio on your machine. The versions of Visual Studio and corresponding Crystal Reports are as follows:

Visual Studio

2005

Edition

Professional, Team System

Crystal Reports

Crystal Reports for Visual Studio 2005

| | | |
|------|--|--|
| 2008 | Professional, Team System | Crystal Reports for Visual Studio 2008 |
| 2010 | Professional, Premium, Ultimate System | Crystal Reports for Visual Studio 2010 |

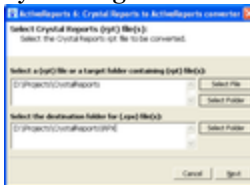
When the Crystal Reports to ActiveReports Converter Wizard is completed, an .rpx file (the ActiveReports layout file) is created. To learn how to add the .rpx file to the existing project, please see **Save and Load Report Layout Files (RPX)**.

To convert a Crystal Reports report into an ActiveReport

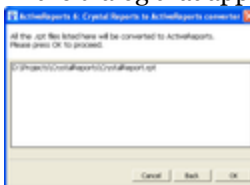
1. From the Start Menu, go to **All Programs > GrapeCity > ActiveReports 6.0 > Tools** and select **Crystal Reports to ActiveReports Converter**.
2. In the ActiveReports 6: Crystal Reports to ActiveReports converter wizard that appears, click the area to proceed to the conversion process.



3. In the dialog that appears, select an rpt file that you want to convert to ActiveReports by using the **Select File** or the **Select Folder** button and click **Next**. You can change the default destination folder for a converted rpx file by clicking the second **Select Folder** button.



4. In the dialog that appears, click **OK**.



Note: The Crystal Reports to ActiveReports Converter converts **XML, SQL and OLE DB (Access, Oracle, SyBase)** data sources from Crystal Reports to ActiveReports. As for other data sources that the Crystal Reports to ActiveReports Converter does not support, you should manually set up these data sources after the conversion is complete - see **DataSource Icon** for more information.

General Limitations

- The controls, functions, and text formats, which are not supported by ActiveReports, are not converted.
- The shadow property of a control is not imported.
- The OLE object is not imported as it is treated as PictureObject in the Crystal Reports object structure.

Converted Report Items

| Crystal Reports | ActiveReports | Note |
|-----------------|---------------|------|
|-----------------|---------------|------|

| | | |
|-----------------------|-----------------------|--|
| BoxObject | Shape | <ul style="list-style-type: none"> • The LineWidth property is not imported. If the box extends to multiple sections, the box is imported as line controls. • The control's z-index is not imported. • The ExtendToBottomOfSection property is not imported. |
| ChartObject | ChartControl | The settings and data are not imported. |
| CrossTabObject | SubReport | Cross tabs are not imported. |
| LineObject | Line | <ul style="list-style-type: none"> • The size of Dot and Dash (the Line Style property) is not exactly the same. • The ExtendToBottomOfSection property is not imported. |
| PictureObject | Picture | Image data is not imported. |
| SubreportObject | SubReport | You need to set the SubReport in code after the conversion. |
| TextObject | Label | <ul style="list-style-type: none"> • Only the functions PageNumber, TotalPageCount and PageNoFM are supported. • The functions PrintDate, PrintTime, ModificationDate, ModificationTime, DataDate, DataTime display System.DateTime.Now.ToString() • The default Alignment is imported as Left. • Some properties like CanGrow, CharacterSpacing, TextRotationAngle and HyperLinkText are not supported. |
| FieldObject | TextBox | <ul style="list-style-type: none"> • Only the functions PageNumber, TotalPageCount and PageNoFM are supported. • The functions PrintDate, PrintTime, ModificationDate, ModificationTime, DataDate, DataTime display System.DateTime.Now.ToString() • The default Alignment is imported as Left. • Some properties like CanGrow, CharacterSpacing, TextRotationAngle and HyperLinkText are not supported. |
| PageHeader/PageFooter | PageHeader/PageFooter | The FieldHeadingObject class is not imported. |

Getting Started

Quickly begin using ActiveReports by reviewing some of the most commonly used features.

This section contains information about:

Adding ActiveReports Controls

Learn how to add ActiveReports controls to the toolbox in Visual Studio.

Adding an ActiveReport to a Project

Learn how to add an ActiveReport to a Visual Studio project.

ActiveReports Templates

Learn about using ActiveReports xml-based and code-based templates in your project.

ActiveReports Designer

Learn what each of the tools and UI items on the report designer can help you to accomplish.

Viewing Reports

Learn how to preview a report at design time as well as how to use the Flash Viewer to view a report at run time.

ActiveReports and the Web

Learn about using ActiveReports on the internet.

Adding ActiveReports Controls

You can **add an ActiveReport to a project** without adding anything to the Visual Studio toolbox, but in order to use the Viewer control, any of the exports, the Designer and related controls, or the WebViewer control, you must first add them to your toolbox.

To add the controls

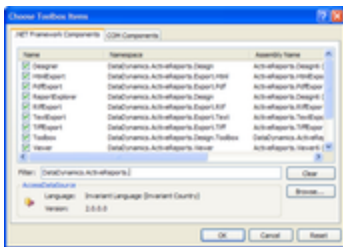
1. Right-click the Visual Studio toolbox tab where you want to add ActiveReports controls and select **Choose Items**.



2. In the Choose Toolbox Items window that appears, on the .NET Framework Components tab, in the Filter textbox, enter **DataDynamics.ActiveReports**.




Tip: Include the dot at the end of **DataDynamics.ActiveReports**. to eliminate the controls you use on the reports themselves and display only the controls that you can use with Windows Forms or Web Forms.



3. Select the check boxes next to any of the controls that you want to add to your toolbox:
 - **Designer**
 - **HtmlExport**
 - PdfExport
 - ReportExplorer
 - RtfExport
 - TextExport
 - TiffExport
 - Toolbox

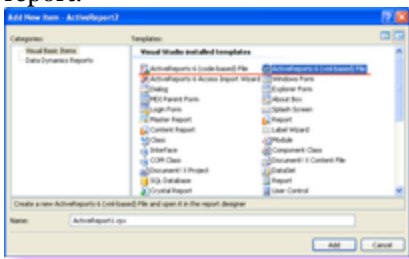
- Viewer
 - **WebViewer**
 - XlsExport
4. For the Silverlight Viewer control, go to the **Silverlight Components** tab and select **Viewer**.
 5. Click **OK** to add the controls to the selected toolbox.

 **Note:** The **Designer**, the **HTML Export** and the **WebViewer** controls require the .NET Framework full profile version. To ensure you are using the full profile version, go to the Visual Studio **Project Properties**, then to the **Compile** tab, **Advanced Compile Options...** (for Visual Basic projects) or to the **Application** tab (for C# projects) and in the **Target framework** field select a full profile version.

Adding an ActiveReport to a Project

To add an ActiveReport to a project

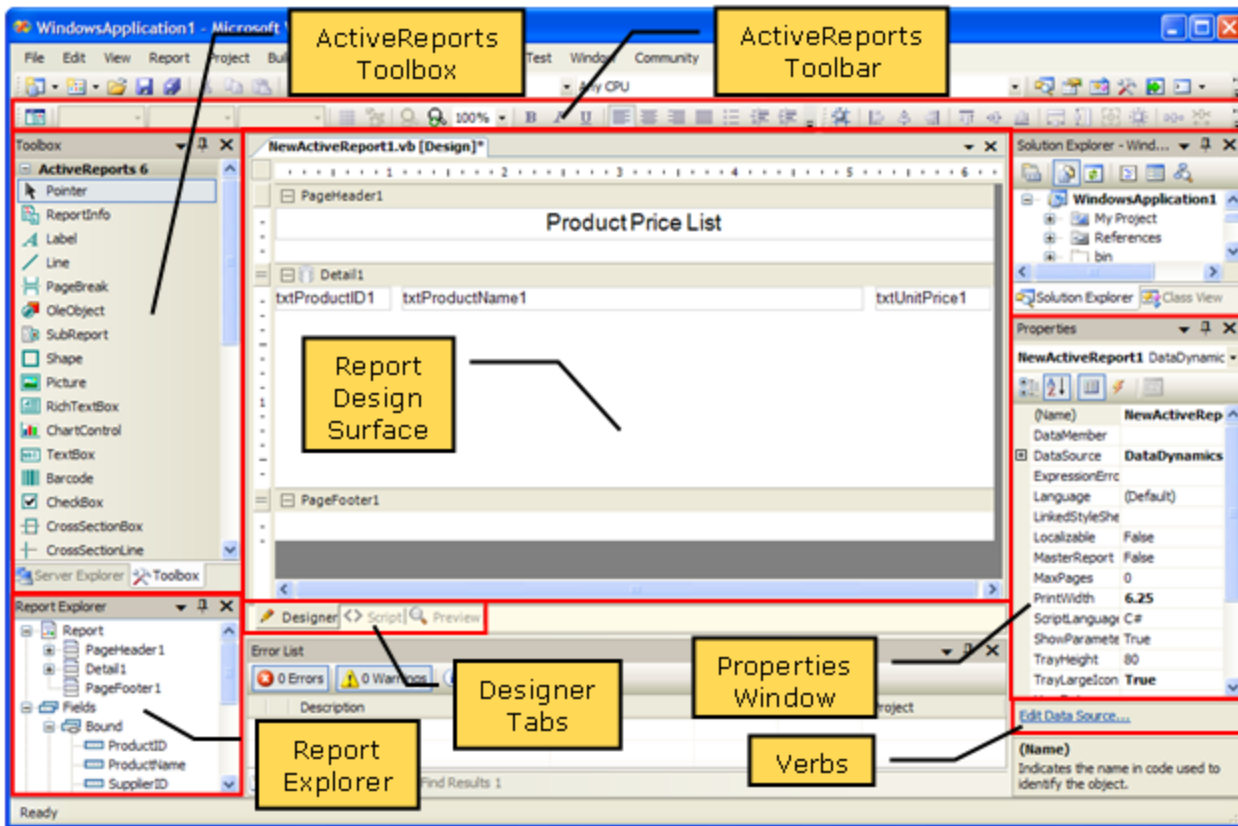
1. From the Visual Studio **Project** menu, select **Add New Item**.
2. Select **ActiveReports 6 (code-based) File** or **ActiveReports 6 (xml-based) File** and name your new report.



3. Click **Add** to add the report to your project and open it in design view.

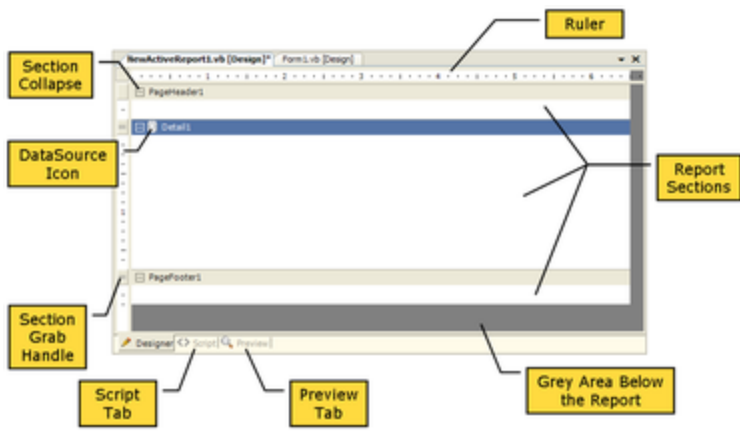
ActiveReports Designer

With its various tools and properties, ActiveReports 6 offers great flexibility in constructing report projects. Click one of the red-outlined areas to view a topic with more information on that section.



Design View

When you first **add an ActiveReport** to a Visual Studio project, the design view of the report displays by default. To reopen one that you have closed, double-click the report in the Solution Explorer. Select any section or the report itself to view available properties in the Properties window. (Click in the grey area below the report to select the report.)



Use the **ruler** to determine how your report will look on paper. If you drag the right edge of the report, or drop controls near the right edge, you can see by the ruler how much the PrintWidth of the report has grown. Keep in mind that you have to add the right and left margin widths to the PrintWidth to determine whether your report will fit on the selected paper size.

By default, a report has three **sections**: a page header, a detail section, and a page footer. Drag controls from the toolbox onto these sections to display your data. Right-click the report and select **Insert** to add other types of header

and footer section pairs. For more information, see **Report Structure**, **Section Events** and **Grouping Data**.

In walkthroughs and how-to topics, you may be told to double-click the **grey area below the report** to create the ReportStart event. You can also click in this area to select the report.

The **preview tab** allows you to check out your report without running the project and displaying it in the Viewer control.

The **script tab** is where you can add VB.NET or C# script for use with RPX portable layout files. For more information, see **Layout Files**.

Use **section grab handles** to drag a section's height up or down.

Click the **data source icon** to open the Report Data Source window, from which you can bind your report to any OLE DB, SQL, or XML data source. For more information, see **DataSource Icon**.

Click a **section collapse** icon to close a section that you have finished working on so you don't accidentally move or change any of your controls.

Report Explorer

In ActiveReports, the Report Explorer gives you a visual overview of the elements that make up the report in the form of a tree view with nodes for:

- The Report
 - Each of its sections
 - Each section's controls
- Fields
 - Bound
 - Calculated
- Parameters
- Report Settings

In the Report Explorer, you can remove individual controls, add parameters and calculated fields, drag bound data fields onto the report as textbox controls, and change report settings. You can also select a section, control, or the report itself to display in the Properties window, where you can modify its properties.

If you do not see the Report Explorer in Visual Studio:

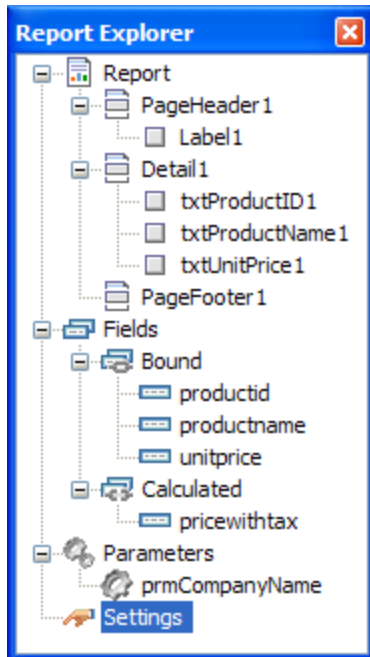
1. Right-click the Visual Studio toolbar and select **ActiveReports 6** to display the designer toolbar.
2. On the designer toolbar, click the **View Report Explorer** button.



Or from the **View** menu, select **Other Windows**, then **Report Explorer**.

When you open the Report Explorer in Visual Studio, it appears every time you create a new Windows Application. You can close it any time.

The Report Explorer lays out all of the elements contained in your report in one place.



The following demonstrates how you can quickly modify a report using the Report Explorer.

To bind data fields to textbox controls

1. In the Report Explorer, select the data field you want to bind to a textbox control.
2. Drag the field onto the design surface of the report. A textbox control is created and bound to the field. The textbox is selected in the Properties window, and the **DataField** and **Name** properties reflect the name of the field. For example, if you dragged a field named **EmployeeID**, the **DataField** property of the textbox is set to **EmployeeID** and the textbox is named **txtEmployeeID1**.

To modify control or section properties

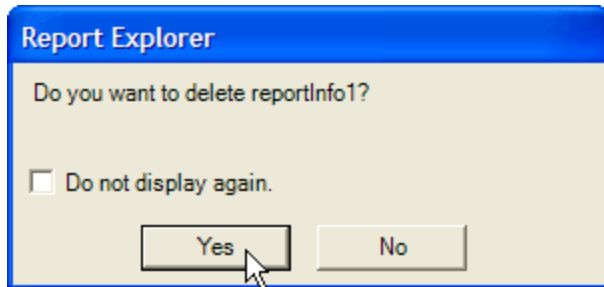
1. In the Report Explorer, select the control or section you want to modify. The Properties window displays all available properties for the item.
2. In the Properties window, set properties as you like.



For more information on controls and some of their properties, see **ActiveReports Toolbox Controls**.

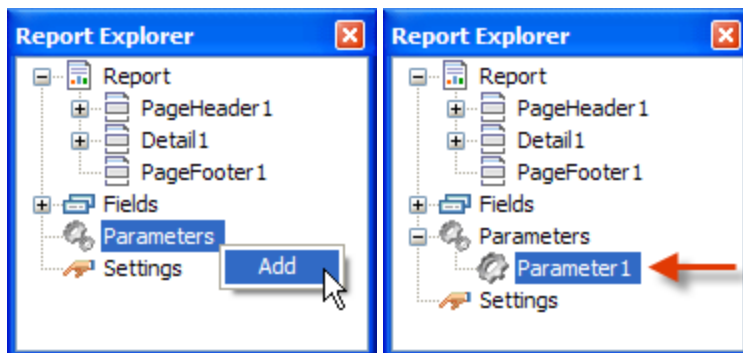
To remove individual controls


1. In the Report Explorer, expand the node that contains the control you want to remove.
2. Right-click the control and select **Delete**.
3. Click **Yes** in the Report Explorer dialog to confirm your decision.



To add parameters

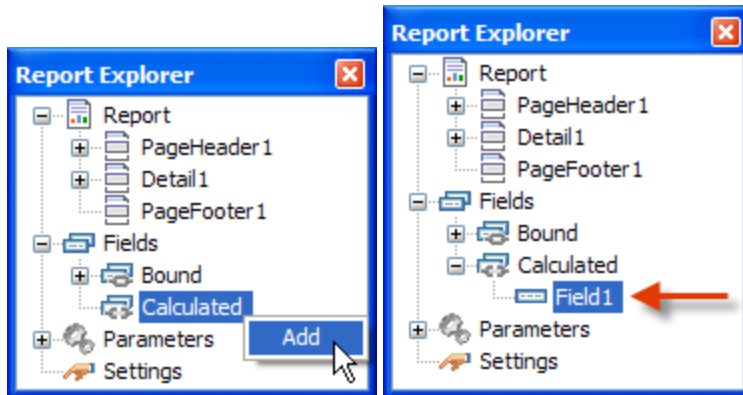
1. In the Report Explorer, right-click the **Parameters** node and select **Add**. The new parameter is displayed in the Report Explorer and in the Properties window.
2. In the Properties window, set the **Prompt** property to a string value to ask users for data.
3. Leave the **PromptUser** property set to **True**. When you run the report, a dialog displays the Prompt to the user.
4. Drag the parameter from the Report Explorer onto the design surface of your report to create a textbox that is bound to the parameter. When you run the report, the value that the user supplies in the prompt dialog displays in the bound textbox on the report.



 For more information, see **Parameters**.

To add calculated fields

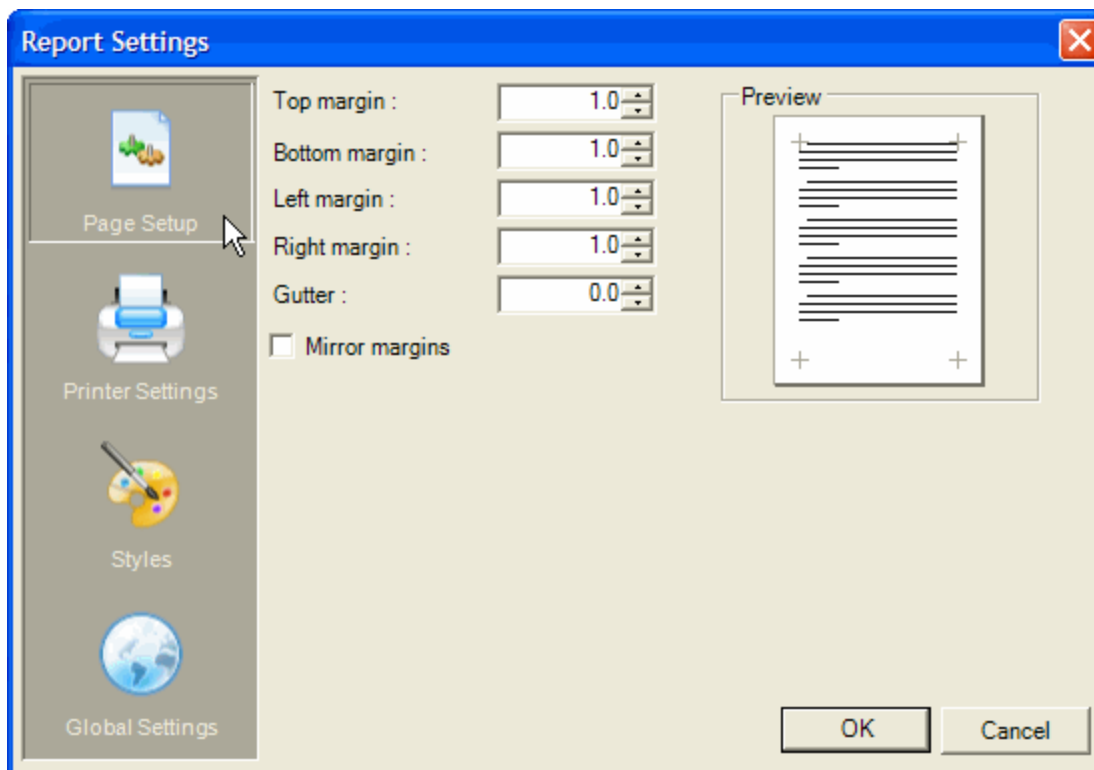
1. In the Report Explorer, expand the **Fields** node.
2. Right-click the **Calculated** node and select **Add**. The new calculated field is displayed in the Report Explorer and in the Properties window.
3. In the Properties window, set the **Formula** property to a calculation, for example: = **unitprice * 1.07**
4. Drag the field from the Report Explorer onto the design surface of your report to create a textbox that is bound to the field.




 For more information, see **Create Calculated Fields**.

To change report settings

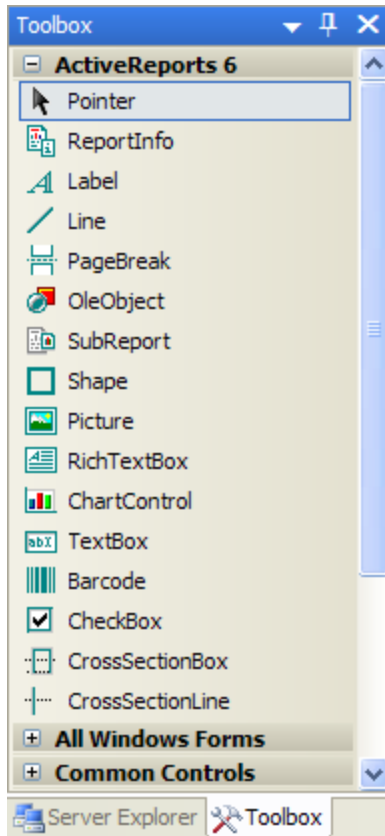
1. In the Report Explorer, right-click the **Settings** node and select **Show** to open the Report Settings dialog.
2. Make changes to properties on the Page Setup, Printer Settings, Styles, and Global Settings tabs.
3. Click **OK** to save the changes.



 For more information, see **Report and Page Settings**.

ActiveReports Toolbox Controls

When you are in design view of an ActiveReport, the Visual Studio toolbox automatically displays the ActiveReports controls that you can drag onto your report.



Tip: If you do not see the Visual Studio toolbox, from the **View** menu, select **Toolbox**.

| Control | Description (Important properties are in bold.) |
|----------------|---|
| Pointer | Selected by default, the pointer allows you to select, move, and resize controls, and resize sections. After you drop or draw a control onto your report, the pointer is automatically selected. |
| ReportInfo | A text box with preset FormatString options, the report info control allows you to quickly display page numbers, page counts, and report dates. For more information, see Display Page Numbers and Report Dates . |
| Label | The label allows you to display static text to describe the data you display in text boxes. Use the Text property to set the label text. Set the Angle property to 900 for displaying text vertically. Now supports direct text input and vertical writing . |
| Line | Use the line to visually separate or call out areas of your report. You can drag it to the size and location you want, or use the X1 , X2 , Y1 , and Y2 properties. The AnchorBottom property lets the line grow along with the section. |
| PageBreak | Use the page break to have the report stop inside a section and resume printing on a new page. You may also wish to use the PageBreakBefore or PageBreakAfter properties available on the sections themselves. |
| OleObject | You can add an OLE object, bound to a database or unbound, directly to your report. When you drop or draw the control onto your report, the Insert Object dialog allows you to create a new |

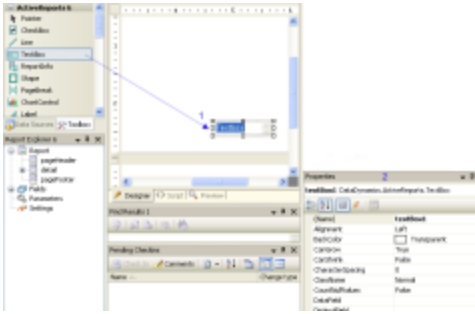
object, or select an existing file.

When you deploy reports that use the OleObject, you must also deploy the ActiveReports.Interop.dll, or for 64-bit machines, the ActiveReports.Interop64.dll.

| | |
|------------------|---|
| SubReport | <p>Use the subreport control as a placeholder for data from a separate report. Use code to connect the separate report to the subreport control.</p> <p>For more information, see Embed Subreports in a Report.</p> |
| Shape | <p>The shape control has a Style property that allows you to select whether to render an ellipse, a rectangle, or a rounded rectangle.</p> |
| Picture | <p>The Image property of the picture control allows you to select any image file to display. Use the PictureAlignment and SizeMode properties to control cropping and alignment.</p> |
| RichTextBox | <p>You can use different formats on adjacent text in the rich text box control, merge data in it, or load an HTML or RTF file into it. Or double-click inside the control to enter text directly into it.</p> <p>For more information, see Load a File into a RichText Control or Mail Merge with RichText.</p> |
| ChartControl | <p>The chart control offers you more than 30 chart types and access to properties controlling every aspect of a chart's appearance.</p> <p>For more information, see Create Charts.</p> |
| TextBox | <p>The text box is the basis of reporting. You can bind it to data using the DataField property, or set it at run time. It is the control that forms when you drag a field onto a report from the Report Explorer. You can use the Summary properties to create summary fields. You can also set the new Padding property to add space between text and the edge of the control.</p> <p>For more information, see Create Summary Fields. Now supports direct text input and vertical writing.</p> |
| Barcode | <p>Select from over 30 barcode Styles, including the latest postal codes. You can bind it to data using the DataField property, and use the Text and CaptionPosition properties to add a caption.</p> <p>For more information, see BarCodes.</p> |
| CheckBox | <p>You can set the Checked property of the check box in code or bind it to a Boolean database value. Provide static text in the Text property.</p> |
| CrossSectionBox | <p>Drag the cross section box onto a header section and it spans any intervening sections to end in the related footer section. Set the Radius property to round the edges of the box.</p> <p>See Cross Section Controls for more information.</p> |
| CrossSectionLine | <p>Drag the cross section line onto a header section and it spans any intervening sections to end in the related footer section. This line control is strictly vertical. If you want a horizontal or diagonal line, use the Line control, which does not span sections.</p> <p>See Cross Section Controls for more information.</p> |

Text Input for TextBox and Label Controls

When working with the **TextBox**, **CheckBox**, and **Label** controls, you can enter or format the text by double-clicking directly in the control area on the report design surface. You can still enter text in the Properties window and in code.



The text in the control is formatted in edit mode by means of the ActiveReports toolbar, or by modifying properties in the Properties window. The formatting commands are applied to the entire text in the selected control.



Note: Text formatting changes in the Properties window are immediately applied to the text in the selected control, and changes made by means of the toolbar are immediately reflected in the Properties window.

TextBox and Label Limitations

- The **Justify** alignment option is not supported in edit mode for the **TextBox** and **Label** controls. To apply this alignment option, a user should exit edit mode, select the **TextBox** or **Label** control by clicking it and then set the alignment to Justify in the Toolbar or in the Properties window.



Note: If the **TextBox** or **Label** control with the Justify alignment is put into edit mode, the alignment value automatically changes to the default one - Left. However, after exiting edit mode, the alignment value of the **TextBox** or **Label** control automatically changes back to Justify.

- The rightmost character in the TextBox control is cut off at runtime and design time if the FontName property is set to a special font, the FontItalic property is set to True and the Alignment property is set to Justify or Right.

TextBox and Label Key Commands

It is also possible to use key commands when working with a textbox or label control in edit mode.

Key Combination

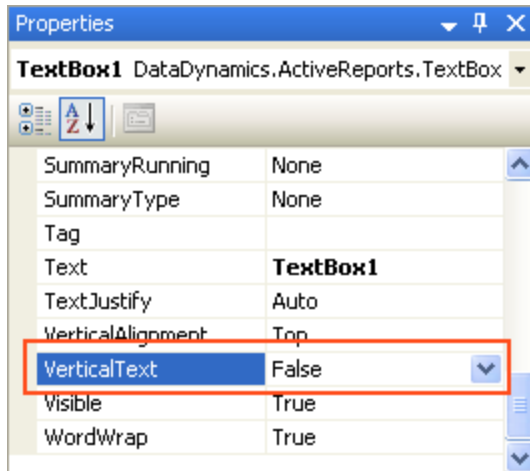
Action

| | |
|---------------|--|
| Enter | New line. |
| Shift + Enter | New line. |
| Ctrl + Enter | New line. |
| Alt + Enter | Saves modifications and exits the edit mode. |
| Esc | Cancels modifications and exits the edit mode. |

In the **End User Designer**, you can disable this feature using the **EditModeEntering** and **EditModeExit** events.

Vertical Writing Support

ActiveReports 6 provides the support of vertical writing that is widely used in East Asian languages. By setting the **VerticalText** property of the Textbox or Label control to True, the text you supply for the controls will be rendered vertically.



Vertical Text Rendering: General Limitations

- Some dividing punctuation marks, middle dots, full stops and commas may render incorrectly, depending on the input mode.

The vertical rendering of the following characters may depend on the input mode

| Character | UCS | Name |
|-----------|------|--|
| !! | 203C | DOUBLE EXCLAMATION MARK |
| □ | 2047 | DOUBLE QUESTION MARK |
| □ | 2048 | QUESTION EXCLAMATION MARK |
| □ | 2049 | EXCLAMATION QUESTION MARK |
| ? | 003F | QUESTION MARK |
| ! | 0021 | EXCLAMATION MARK |
| ; | 003B | SEMICOLON Remark: used horizontal composition |
| . | 002E | FULL STOP Remark: used horizontal composition |
| , | 002C | COMMA Remark: used horizontal composition |

- The Unicode ideographic characters 3013 ("geta mark") and 261E ("white right pointing index") are rendered incorrectly in vertical mode if the Font property is set to Meiryo.
- The following Hiragana and Katakana *character combinations* are rendered as separate characters in vertical mode (the **VerticalText** property is set to True), instead of being rendered as one character:

Hiragana and Katakana combinations that have rendering limitations in vertical mode

| Character | UCS | Name |
|-----------|-----|------|
|-----------|-----|------|

| | | |
|-----|--------------|--|
| か ° | <304B, 309A> | <HIRAGANA LETTER KA, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| き ° | <304D, 309A> | <HIRAGANA LETTER KI, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| く ° | <304F, 309A> | <HIRAGANA LETTER KU, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| け ° | <3051, 309A> | <HIRAGANA LETTER KE, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| こ ° | <3053, 309A> | <HIRAGANA LETTER KO, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| カ ° | <30AB, 309A> | <KATAKANA LETTER KA, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| キ ° | <30AD, 309A> | <KATAKANA LETTER KI, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| ク ° | <30AF, 309A> | <KATAKANA LETTER KU, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| ケ ° | <30B1, 309A> | <KATAKANA LETTER KE, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| コ ° | <30B3, 309A> | <KATAKANA LETTER KO, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| セ ° | <30BB, 309A> | <KATAKANA LETTER SE, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| ツ ° | <30C4, 309A> | <KATAKANA LETTER TU, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |
| ト ° | <30C8, 309A> | <KATAKANA LETTER TO, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |

Vertical Text Rendering: Viewer Limitations

- Small kanas are rendered incorrectly if the Angle property of the Textbox or Label control is set to 90 degrees

Small kanas that are rendered incorrectly in the Viewer control if rotated 90 degrees

| Character | UCS | Name |
|-----------|------|--------------------------|
| □ | 31F0 | KATAKANA LETTER SMALL KU |
| □ | 31F1 | KATAKANA LETTER SMALL SI |
| □ | 31F2 | KATAKANA LETTER SMALL SU |
| □ | 31F3 | KATAKANA LETTER SMALL TO |
| □ | 31F4 | KATAKANA LETTER SMALL NU |

| | | |
|--------------------------|--------------|--|
| <input type="checkbox"/> | 31F5 | KATAKANA LETTER SMALL HA |
| <input type="checkbox"/> | 31F6 | KATAKANA LETTER SMALL HI |
| <input type="checkbox"/> | 31F7 | KATAKANA LETTER SMALL HU |
| <input type="checkbox"/> | 31F8 | KATAKANA LETTER SMALL HE |
| <input type="checkbox"/> | 31F9 | KATAKANA LETTER SMALL HO |
| <input type="checkbox"/> | 31FA | KATAKANA LETTER SMALL MU |
| <input type="checkbox"/> | 31FB | KATAKANA LETTER SMALL RA |
| <input type="checkbox"/> | 31FC | KATAKANA LETTER SMALL RI |
| <input type="checkbox"/> | 31FD | KATAKANA LETTER SMALL RU |
| <input type="checkbox"/> | 31FE | KATAKANA LETTER SMALL RE |
| <input type="checkbox"/> | 31FF | KATAKANA LETTER SMALL RO |
| <input type="checkbox"/> | <31F7, 309A> | <KATAKANA LETTER SMALL HU, COMBINING KATAKANA-HIRAGANA SEMI-VOICED SOUND MARK> |

Vertical Text Rendering: FlashViewer Limitations

- The Unicode ideographic character 00A7 ("section sign") is cut off when rendered in vertical mode.
- The open square bracket symbol is cutoff when rendered in vertical mode if the Font property is set to MS P Gothic, MS UI Gothic or MS P Mincho.
- Any character is rendered incorrectly in vertical mode if the Italic property is set to True for the Meiryo font.
- The Unicode ideographic characters 2985 ("left white parenthesis") and 2986 ("right white parenthesis") are rendered incorrectly in vertical mode if the Font property is set to Meiryo.
- FlashViewer renders vertical text in the Textbox or Lable control of any width as one line.
- FlashViewer does not render correctly some ideographic characters (cl-19).

Ideographic characters (cl-19) that are rendered incorrectly in FlashViewer

| Character | UCS | Name |
|-----------|------|------------------------------|
| ♂ | 2642 | MALE SIGN |
| ♀ | 2640 | FEMALE SIGN |
| ☆ | 2606 | WHITE STAR |
| ★ | 2605 | BLACK STAR |
| △ | 25B3 | WHITE UP-POINTING TRIANGLE |
| ▲ | 25B2 | BLACK UP-POINTING TRIANGLE |
| ▽ | 25BD | WHITE DOWN-POINTING TRIANGLE |

| | | |
|---|------|--------------------------------------|
| ▼ | 25BC | BLACK DOWN-POINTING TRIANGLE |
| ▷ | 25B7 | WHITE RIGHT-POINTING TRIANGLE |
| ▶ | 28B6 | BLACK RIGHT-POINTING TRIANGLE |
| ◁ | 25C1 | WHITE LEFT-POINTING TRIANGLE |
| ◀ | 25C0 | BLACK LEFT-POINTING TRIANGLE |
| # | 266F | MUSIC SHARP SIGN |
| b | 266D | MUSIC FLAT SIGN |
| ♪ | 266A | EIGHTH NOTE |
| ♮ | 266E | MUSIC NATURAL SIGN |
| ♪ | 266B | BEAMED EIGHTH NOTES |
| ♪ | 266C | BEAMED SIXTEENTH NOTES |
| ♪ | 2669 | QUARTER NOTE |
| ✓ | 2713 | CHECK MARK |
| ◐ | 25D0 | CIRCLE WITH LEFT HALF BLACK |
| ◑ | 25D1 | CIRCLE WITH RIGHT HALF BLACK |
| ◒ | 25D2 | CIRCLE WITH LOWER HALF BLACK |
| ◓ | 25D3 | CIRCLE WITH UPPER HALF BLACK |
| ❶ | 2776 | DINGBAT NEGATIVE CIRCLED DIGIT ONE |
| ❷ | 2777 | DINGBAT NEGATIVE CIRCLED DIGIT TWO |
| ❸ | 2778 | DINGBAT NEGATIVE CIRCLED DIGIT THREE |
| ❹ | 2779 | DINGBAT NEGATIVE CIRCLED DIGIT FOUR |
| ❺ | 277A | DINGBAT NEGATIVE CIRCLED DIGIT FIVE |
| ❻ | 277B | DINGBAT NEGATIVE CIRCLED DIGIT SIX |
| ❼ | 277C | DINGBAT NEGATIVE CIRCLED DIGIT SEVEN |
| ❽ | 277D | DINGBAT NEGATIVE CIRCLED DIGIT EIGHT |
| ❾ | 277E | DINGBAT NEGATIVE CIRCLED DIGIT NINE |
| ❿ | 277F | DINGBAT NEGATIVE CIRCLED NUMBER TEN |
| ◻ | 24EB | NEGATIVE CIRCLED NUMBER ELEVEN |

| | | |
|---|------|-----------------------------------|
| ☐ | 24EC | NEGATIVE CIRCLED NUMBER TWELVE |
| ☐ | 24ED | NEGATIVE CIRCLED NUMBER THIRTEEN |
| ☐ | 24EE | NEGATIVE CIRCLED NUMBER FOURTEEN |
| ☐ | 24EF | NEGATIVE CIRCLED NUMBER FIFTEEN |
| ☐ | 24F0 | NEGATIVE CIRCLED NUMBER SIXTEEN |
| ☐ | 24F1 | NEGATIVE CIRCLED NUMBER SEVENTEEN |
| ☐ | 24F2 | NEGATIVE CIRCLED NUMBER EIGHTEEN |
| ☐ | 24F3 | NEGATIVE CIRCLED NUMBER NINETEEN |
| ☐ | 24F4 | NEGATIVE CIRCLED NUMBER TWENTY |
| Ⓐ | 24D0 | CIRCLED LATIN SMALL LETTER A |
| Ⓑ | 24D1 | CIRCLED LATIN SMALL LETTER B |
| Ⓒ | 24D2 | CIRCLED LATIN SMALL LETTER C |
| Ⓓ | 24D3 | CIRCLED LATIN SMALL LETTER D |
| Ⓔ | 24D4 | CIRCLED LATIN SMALL LETTER E |
| Ⓕ | 24D5 | CIRCLED LATIN SMALL LETTER F |
| Ⓖ | 24D6 | CIRCLED LATIN SMALL LETTER G |
| Ⓗ | 24D7 | CIRCLED LATIN SMALL LETTER H |
| Ⓘ | 24D8 | CIRCLED LATIN SMALL LETTER I |
| ⓫ | 24D9 | CIRCLED LATIN SMALL LETTER J |
| ⓬ | 24DA | CIRCLED LATIN SMALL LETTER K |
| ⓭ | 24DB | CIRCLED LATIN SMALL LETTER L |
| ⓮ | 24DC | CIRCLED LATIN SMALL LETTER M |
| ⓯ | 24DD | CIRCLED LATIN SMALL LETTER N |
| ⓰ | 24DE | CIRCLED LATIN SMALL LETTER O |
| ⓱ | 24DF | CIRCLED LATIN SMALL LETTER P |
| ⓲ | 24E0 | CIRCLED LATIN SMALL LETTER Q |
| ⓳ | 24E1 | CIRCLED LATIN SMALL LETTER R |
| ⓴ | 24E2 | CIRCLED LATIN SMALL LETTER S |

| | | |
|---|------|------------------------------|
| Ⓣ | 24E3 | CIRCLED LATIN SMALL LETTER T |
| Ⓤ | 24E4 | CIRCLED LATIN SMALL LETTER U |
| Ⓥ | 24E5 | CIRCLED LATIN SMALL LETTER V |
| Ⓦ | 24E6 | CIRCLED LATIN SMALL LETTER W |
| Ⓧ | 24E7 | CIRCLED LATIN SMALL LETTER X |
| Ⓨ | 24E8 | CIRCLED LATIN SMALL LETTER Y |
| Ⓩ | 24E9 | CIRCLED LATIN SMALL LETTER Z |
| ① | 2460 | CIRCLED DIGIT ONE |
| ② | 2461 | CIRCLED DIGIT TWO |
| ③ | 2462 | CIRCLED DIGIT THREE |
| ④ | 2463 | CIRCLED DIGIT FOUR |
| ⑤ | 2464 | CIRCLED DIGIT FIVE |
| ⑥ | 2465 | CIRCLED DIGIT SIX |
| ⑦ | 2466 | CIRCLED DIGIT SEVEN |
| ⑧ | 2467 | CIRCLED DIGIT EIGHT |
| ⑨ | 2468 | CIRCLED DIGIT NINE |
| ⑩ | 2469 | CIRCLED NUMBER TEN |
| ⑪ | 246A | CIRCLED NUMBER ELEVEN |
| ⑫ | 246B | CIRCLED NUMBER TWELVE |
| ⑬ | 246C | CIRCLED NUMBER THIRTEEN |
| ⑭ | 246D | CIRCLED NUMBER FOURTEEN |
| ⑮ | 246E | CIRCLED NUMBER FIFTEEN |
| ⑯ | 246F | CIRCLED NUMBER SIXTEEN |
| ⑰ | 2470 | CIRCLED NUMBER SEVENTEEN |
| ⑱ | 2471 | CIRCLED NUMBER EIGHTEEN |
| ⑲ | 2472 | CIRCLED NUMBER NINETEEN |
| ⑳ | 2473 | CIRCLED NUMBER TWENTY |
| ♠ | 2664 | WHITE SPADE SUIT |

| | | |
|---|------|---------------------|
| ♠ | 2660 | BLACK SPADE SUIT |
| ◇ | 2662 | WHITE DIAMOND SUIT |
| ♠ | 2666 | BLACK DIAMOND SUIT |
| ♥ | 2661 | WHITE HEART SUIT |
| ♥ | 2665 | BLACK HEART SUIT |
| ♣ | 2667 | WHITE CLUB SUIT |
| ♣ | 2663 | BLACK CLUB SUIT |
| □ | 2616 | WHITE SHOGI PIECE |
| □ | 2617 | BLACK SHOGI PIECE |
| ☎ | 3020 | POSTAL MARK FACE |
| ☎ | 260E | BLACK TELEPHONE |
| ☀ | 2600 | BLACK SUN WITH RAYS |
| ☁ | 2601 | CLOUD |
| ☂ | 2602 | UMBRELLA |
| ☺ | 2603 | SNOWMAN |
| ♨ | 2668 | HOT SPRINGS |
| ▭ | 25B1 | WHITE PARALLELOGRAM |

Vertical Text Export Options

Vertical writing is supported for the following export filters:

- **PDF**

Limitations:

1. If the width of the TextBox or Label control, containing vertical text, is larger than a single character, the text in the exported PDF file will overlap.
2. Vertical text in the Label control with the **Angle ('Angle Property' in the on-line documentation)** property set to a value other than 0 will not display correctly in the resulting PDF file.
3. Open, curly and square brackets with the Italic property set to True are rendered incorrectly in the resulting PDF file.
4. Some Japanese Kanji characters with the Italic property set to True are rendered incorrectly in vertical mode in the resulting PDF file.


- **TIFF**
- **HTML**

Limitations:

1. In case of the HTML export filter, the vertical text is supported just for the DynamicHtml output type and works with Microsoft Internet Explorer only.

Cross Section Controls

ActiveReports includes two new cross section controls that you can drop into any header or footer section. The controls automatically span any intervening sections to end in the related footer or header section.

 **Note:** If you try to drop a cross-section control into a section other than a header or footer, the mouse pointer changes to Unavailable, and you cannot drop the control.



CrossSectionLine



The CrossSectionLine control is a vertical line that begins in the GroupHeader and ends in the corresponding section footer. At run time, the line stretches across the detail section. You can control the appearance of the line with properties such as:


- **LineColor** allows you to select a color for the line.
- **LineStyle** allows you to select from various styles of dashes, dots, or solid.
- **LineWeight** allows you to set the width of the line in pixels.

CrossSectionBox

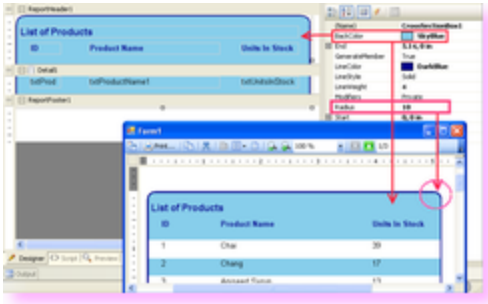


The CrossSectionBox control draws a rectangle that begins in a section header and ends in the corresponding section footer. You can control the appearance of the rectangle with the properties above, plus:

- **Radius** allows you to set a value in pixels to round the corners of the box, where 0 is a rectangle and 200 is a circle.
- **BackColor** allows you to select a background color for the control.

 **Note:** The background color of the CrossSectionBox control is rendered on the report page before any other sections and controls. This means that the background color is visible on transparent sections only.

Unlike the background color, the border of the CrossSectionBox control is rendered after all other controls on the page.



Feature Limitation

The Cross Section controls will not be rendered when used in the Group section of a multi-column report (the **ColumnCount** ('**ColumnCount Property**' in the on-line documentation) property of the Detail section) if the **ColumnLayout** ('**ColumnLayout Property**' in the on-line documentation) property of the Group section is set to True.

Toolbar

You can rearrange buttons and menu options, as well as hide, display, dock or float the ActiveReports toolbar in Visual Studio.

To access the toolbars shortcut menu

Right-click anywhere in the toolbar area.

The shortcut 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.



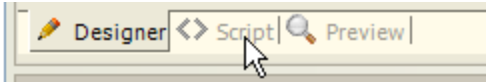
The ActiveReports toolbar is made up of the following components:

- **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 text in the RichText control, or all text in any other control.
- **Size** Sets the font size of the selected text in the RichText control, or all text in any other control.
- **View Grid** Indicates whether the grid display is on or off. If the grid is on, **snap lines** are turned off.
- **Reorder Groups** Opens the Group Order dialog, where you can drag and drop groups to rearrange them. Enabled when you have **multiple groups**.
- **Zoom Out** Reduces the magnification level of the design surface.
- **Zoom In** Increases the magnification level of the design surface.
- **Zoom** Sets the magnification level of the design surface between 100 and 800%.
- **Bold** Sets or removes text emphasis. Applies to selected text in the RichText control, or all text in any other control.
- **Italic** Sets or removes text slant. Applies to selected text in the RichText control, or all text in any other control.
- **Underline** Sets or removes text underlining. Applies to selected text in the RichText control, or all text in any other control.
- **Align Left** Aligns the text left in the control area.
- **Align Center** Centers the text in the control area.
- **Align Right** Aligns the text right in the control area.
- **Justify** Fully justifies the text in the control area.
- **Bullets** Adds bullets to selected text in the RichText control area.

- **Decrease Indent** Decreases the indent of selected text in the RichText control area.
- **Increase Indent** Increases the indent of selected text in the RichText control area.

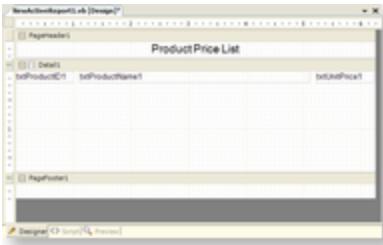
Designer Tabs

The designer tabs located at the bottom of the ActiveReports design surface allow you to quickly access three aspects of ActiveReports: the Designer, the Script, and the Preview.



Designer

The designer tab is selected by default when you create or open an ActiveReport in Visual Studio. On this tab, you can drag controls from the toolbox to create a layout, add, remove, and resize sections, set properties on sections and controls, bind the report to a data source, and create event-handling methods.

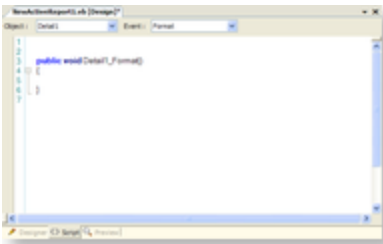


With ActiveReports Professional Edition, you can create a similar designer for end users with the **Designer Control**.

Script

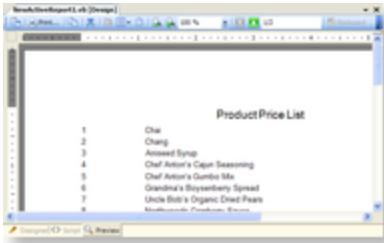
Select the Script tab to open the script editor, where you can add portable code to reports that you want to save as RPX layouts. For more information, see **Layout Files**.

The script editor contains two drop-down boxes that allow you to select any section of the ActiveReport and any events associated with that section, or the report itself and related events. When you select an event, the script editor generates a method stub for the event.



Preview

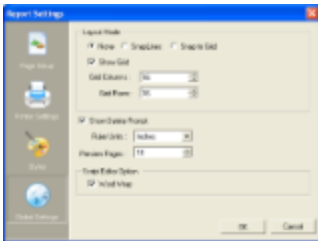
The Preview tab allows you to quickly view your report at run time without the need to actually run your project. This makes it easy to quickly see the run-time impact of changes you make in the designer or the code.



Snap Lines

By default, ActiveReports now uses snap lines instead of a grid on the design surface. You can use snap lines in the ActiveReport design view within Visual Studio, and your end users can use them in the compiled End User Designer.

Snap lines are dynamic horizontal and vertical layout guidelines used to make it easier to position controls on your reports, and are similar to the ones found in Visual Studio 2005 and later. If you prefer to use a grid, you can change this setting in the Report Settings window on the Global Settings tab.



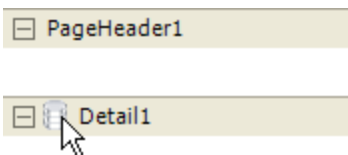
When you drag a control around on the report, blue snap lines appear and the control slows down when the control aligns with another control or a section edge, similar to a magnet pulling the control into alignment. Snap lines even show you when your control is aligned with controls in other sections. Unlike using a grid, the control moves freely around the report and you can place it anywhere.



Tip: If you plan to export a report to Excel format, use snap lines to ensure that your controls are aligned in columns and rows to avoid empty cells in the spreadsheet.

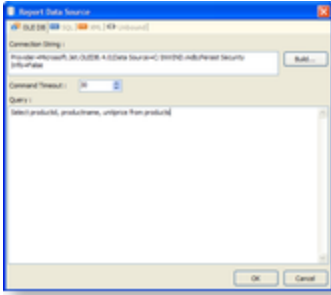
DataSource Icon

Use the DataSource icon to bind your report to a data source at design time.



When you click the icon, the Report Data Source window appears. You can connect the report to OLE DB, SQL, or XML data, and supply a query to retrieve the data you want. You can also add parameters to the report by using parameter

syntax in the SQL query. For more information, see **Add Parameters**.



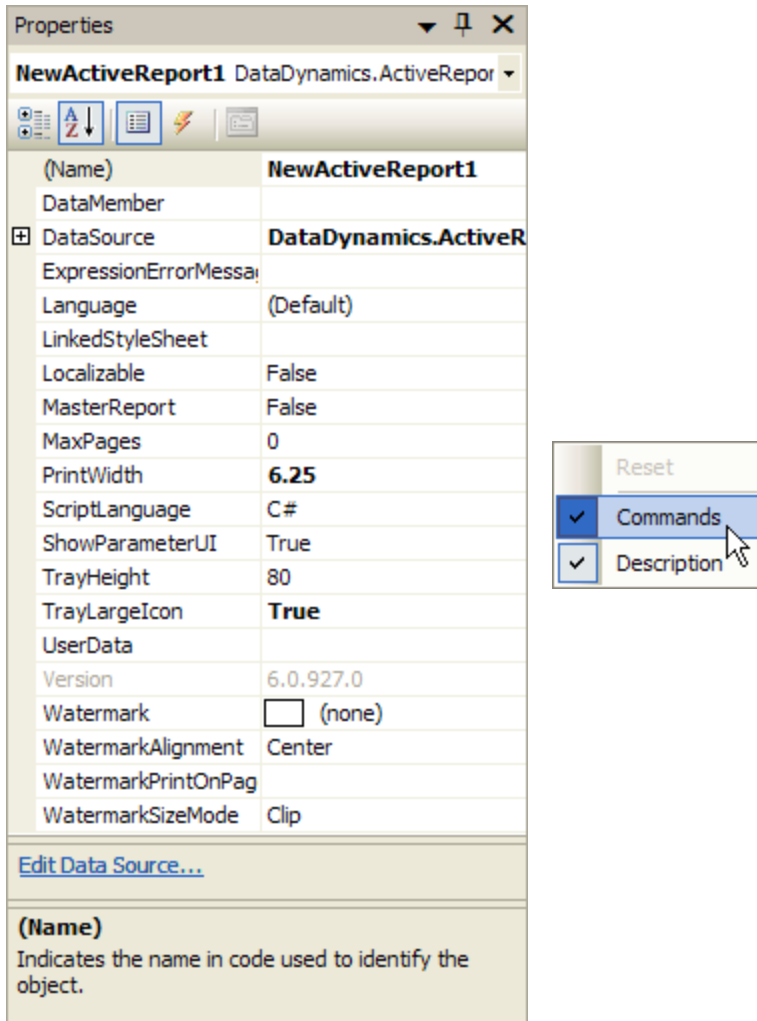
Select the OLE DB, SQL, or XML tab to see the options you have for each type of data source. For more information, see **Bind Reports to a Data Source**.

Properties Window

The Visual Studio Properties window is an important tool when you are designing an ActiveReports layout. Select any control, section, or the report itself to gain access to its properties in the Properties window.

Select a property to reveal a description in the bottom section of the window. Just above the description is a commands section that contains verbs, links to windows that give you access to further properties for the item. Only the chart control and the report itself have associated verbs.

If you cannot see the Description or Command section, right-click anywhere on the Properties window and ensure that both are selected, or try resizing the sections.



The screenshot shows the 'Properties' window for 'NewActiveReport1'. The 'DataSource' property is set to 'DataDynamics.ActiveR'. The 'PrintWidth' property is set to '6.25'. The 'ScriptLanguage' property is set to 'C#'. The 'ShowParameterUI' property is set to 'True'. The 'TrayHeight' property is set to '80'. The 'TrayLargeIcon' property is set to 'True'. The 'Version' property is set to '6.0.927.0'. The 'Watermark' property is set to '(none)'. The 'WatermarkAlignment' property is set to 'Center'. The 'WatermarkPrintOnPag' property is set to 'Clip'. The 'WatermarkSizeMode' property is set to 'Clip'. A context menu is open over the 'Commands' property, showing options for 'Reset', 'Commands', and 'Description'.

| (Name) | NewActiveReport1 |
|------------------------|----------------------|
| DataMember | |
| DataSource | DataDynamics.ActiveR |
| ExpressionErrorMessage | |
| Language | (Default) |
| LinkedStyleSheet | |
| Localizable | False |
| MasterReport | False |
| MaxPages | 0 |
| PrintWidth | 6.25 |
| ScriptLanguage | C# |
| ShowParameterUI | True |
| TrayHeight | 80 |
| TrayLargeIcon | True |
| UserData | |
| Version | 6.0.927.0 |
| Watermark | (none) |
| WatermarkAlignment | Center |
| WatermarkPrintOnPag | Clip |
| WatermarkSizeMode | Clip |

[Edit Data Source...](#)

(Name)
Indicates the name in code used to identify the object.

For more information on some of the important properties for ActiveReports controls, see **ActiveReports Toolbox Controls**.

ActiveReports Templates

To create a report, a user must select a template containing the report layout. In ActiveReports 6, there are two types of such templates – the XML-based report template, **ActiveReports 6 (xml-based) File**, and the code-based report template, **ActiveReports 6 (code-based) File**. It is possible to use both types of report templates within one project.

A report layout based on the code-based template is saved as a C# or Visual Basic for .NET file, whereas a report layout based on the xml-based template is saved as an RPX file.

The RPX file is an XML-formatted file which contains the layout information and any scripts added to the report. RPX files with scripting allow to change and modify distributed reports without recompiling the project. They also make it possible to use a database of report file names to set up a collection of reports to run. You can use an RPX file with scripting as a stand-alone file in a web project. For the information on how to add script to an xml-based report, please refer to **Adding Script to an XML-Based Report**.

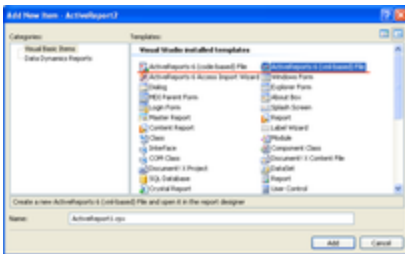
Note: An RPX file in ActiveReports for .NET 2.0 has an associated .cs or .vb file containing code added to the report, whereas an RPX file in ActiveReports 6 does not require any associated code file. For additional information, see **ActiveReports for .NET 2.0 Side-by-Side Installation**.

To convert a code-based report layout to RPX

1. From the **Report** menu, select **Save Layout**.
2. In the Save dialog that appears, enter a name for the report, i.e. rptScript.rpx, and click the **Save** button.
3. In a Visual Studio project, select an xml-based report template (**ActiveReports 6 (xml-based) File in Project>Add New Item**).
4. Click on **File > Open > File...**
5. Select the RPX report layout from the appropriate location.
6. Click **Open** to load the report layout.

To select a report template

To select a template for a report, select **Add New Item** in the Visual Studio **Project** menu, select the template for a new report in the **Add New Item** dialog and then click the **Add** button:



Once a template is selected, the process of designing a report is similar for both types of report templates - see the walkthroughs **Basic Data Bound Reports**, **Basic XML-Based Reports (RPX)**. With the xml-based report template, a user cannot use regular code. Instead, a user is able to access the controls and sections in script editor by using "this"(c#) or "Me"(vb) in addition to the current way of using "rpt" (see **Scripting** for more details).



Note: Since the RPX file can be read with any text editor, the **AddCode** ('AddCode Method' in the on-line documentation) or **AddNamedItem** ('AddNamedItem Method' in the on-line documentation) method should be used to add secure information, such as a connection string, to a project.

Adding Script to an XML-Based Report

To learn how to design a simple report based on the xml-based template, refer to the walkthrough **Basic XML-Based Reports (RPX)**.

When adding script to an xml-based report, the following options are available:

- **The use of "this" (as in C# code-behind) or "Me" (as in VB code-behind) to reference the report.**

With the xml-based report template, a user cannot use regular code. Instead, a user is able to access the controls and sections in script editor by using "this"(c#) or "Me"(vb) in addition to the current way of using "rpt" (see **Scripting** for more details).



- **Intellisense support**

The ActiveReports script editor supports IntelliSense that helps insert the language elements and provides other helpful options when adding script to a report.



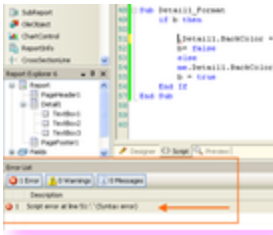
- **Run-time error handling**

When run-time errors occur, a corresponding error message is displayed under the Preview tab with the indication of the problem.



- **Errors List**

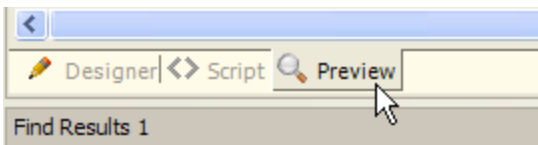
Syntax and other types of errors are displayed in the Errors List below the ActiveReports Designer.



Viewing Reports

Previewing Reports at Design Time

ActiveReports makes it easy for you to preview your report while you are still creating it. Just click the **Preview** tab at the bottom of the ActiveReport designer. In this way you can see and work with the report without the need to run the project.



Using the ActiveReports Windows Form Viewer

In this example, the report is named **rptMain**. If you copy and paste the code below, replace **rptMain** with the name of your report.

To display report output in the Viewer control

1. From the Visual Studio toolbox, drag the ActiveReports Viewer control onto your Windows form.
2. Set the viewer's **Dock** property to **Fill**.
3. Double-click the title bar of the Windows Form to create an event-handling method for the form **Load** event.

4. Add code to the handler to run the report and display it in the viewer.

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET


Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New rptMain()
rpt.Run()
Viewer1.Document = rpt.Document
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptMain rpt = new rptMain();
rpt.Run();
this.viewer1.Document = rpt.Document;
```

 **Note:** To enable the viewer's **Copy** button, add references to the **RtfExport** and **TextExport** DLLs.

Windows Form Viewer Hot Keys And Shortcuts

The following shortcuts are available on the Windows Form viewer:

| Keyboard Shortcut | Action |
|---------------------------|--|
| Ctrl + F | Shows the find dialog. |
| Ctrl + P | Shows the print dialog. |
| Esc | Closes the find or print dialogs. |
| Page Down | Moves to the next page. |
| Page Up | Moves to the previous page. |
| Ctrl + T | Shows or hides the table of contents. |
| Ctrl + Home | Moves to the first page. |
| Ctrl + End | Moves to the last page. |
| Ctrl + Right | Navigates forward. |
| Ctrl + Left | Navigates backward. |
| Ctrl + - | Zooms out. |
| Ctrl + + | Zooms in. |
| Left, Right, Up, Down | Moves the visible area of the page in the corresponding direction. |
| Ctrl + 0 (zero) | Sets the zoom level to 100%. |
| Ctrl + rotate mouse wheel | Changes the zoom level up or down. |
| Ctrl + M | Turns on the Continuous scroll mode. |
| Ctrl + S | Turns off the Continuous scroll mode. |

The following shortcuts are available in the Thumbnails pane of the Windows Form Viewer sidebar.

| Keyboard Shortcut | Action |
|--------------------------|---------------|
|--------------------------|---------------|

| | |
|-----------|---|
| Up | Selects the above page. |
| Down | Selects the below page. |
| Left | Selects the left page. If there is no left page, the last page of the previous row is selected. |
| Right | Selects the right page. If there is no right page, the first page of the next row is selected. |
| Page Up | Moves to the previous Thumbnail view. |
| Page Down | Moves to the next Thumbnail view. |
| Home | Selects the first page. |
| End | Selects the end page. |

ActiveReports and the Web

Professional Edition

The ActiveReports Professional Edition license entitles you to use the WebViewer, which allows you to quickly display reports in any of four viewer types: **HtmlViewer**, **RawHtml**, **AcrobatReader**, or the new **FlashViewer**.

Before using the WebViewer, you must first **configure the HTTPHandlers**.

Getting Started with the Web Viewer (Pro Edition)

Web Viewer Print Options

Flash Viewer Options

Flash Viewer Hot Keys and Shortcuts

Silverlight Viewer (Pro Edition)

Silverlight Viewer Hot Keys and Shortcuts (Pro Edition)

Standard Edition

The Standard Edition license does not have a Web viewer, but you can export reports for use on the Web or use Web Services to distribute documents or data sources. For more information, see **Web Walkthroughs (Standard Edition)**.

Medium Trust Support

Windows Azure Support

Getting Started with the Web Viewer (Pro Edition)

The WebViewer control allows you to quickly display reports in Web applications, and now includes a **FlashViewer** option. You must purchase the Professional Edition license in order to use the WebViewer control; the FlashViewer can be used in both Standard and Pro Editions. If you have the Standard Edition license, see **Web Walkthroughs (Standard Edition)**.

To use the WebViewer control, you must first add it to the Visual Studio toolbox. See **Adding ActiveReports Controls** for more information.

You must also **configure HTTP Handlers** on your server so that IIS knows how to associate ActiveReports files in the browser.

Once you have the control in your toolbar, you can add it to the **Design** view of an ASPX page in a Web application, and

set its **ReportName** property to the name of an ActiveReport within your solution.



Important: If you elect to use the FlashViewer ViewerType, you must copy the **ActiveRepor.FlashViewer.swf** file into your project folder. This file is located in:

C:\Program Files\GrapeCity\ActiveReports 6\Deployment (on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment).

For more control of the output, the WebViewer offers many properties, some of which only apply when you select a specific **ViewerType**. Drop down the table below for more information on each of the properties.

WebViewer Control Properties

| Property Name | Description |
|---|--|
| New FlashViewerOptions These options apply only when you select the FlashViewer ViewerType. | <ul style="list-style-type: none"> • Display Transparency Allows you to specify whether to print transparent objects. • HyperLinkBackColor Allows you to specify the background color of the controls that host a hyperlink. • HyperLinkForeColor Allows you to specify the color of the text within a control that hosts a hyperlink. • HyperLinkUnderline Determines whether the text in the control that hosts a hyperlink is underlined. • MultiPageViewColumns Determines the amount of columns to show when ViewType is MultiPage. • MultiPageViewRows Determines the amount of rows to show when the ViewType is MultiPage. • PageNumber Allows you to specify the page to display initially. • PrintOptions Allows you to specify how the viewer handles page orientation and scaling. The StartPrint option allows you to specify whether to print the report after loading for one-touch printing. If you set the WebViewer's Height and Width properties to 0, you can have the report print without displaying it. • ResourceLocale Allows you to specify the culture for localization. Separate multiple values with commas. • ResourceUrl Allows you to specify the comma-separated list of URLs to SWF files with resource files. • SearchResultsBackColor Allows you to specify the background color of the highlighted text in the Find dialog of the WebViewer control. • SearchResultsForeColor Allows you to specify the color of the highlighted text in the Find dialog of the WebViewer control. • ShowSplitter Allows you to specify whether to display the splitter, which allows the user to compare report pages in the viewer. • ThemeUrl Allows you to specify the relative URL of a skin to use on the FlashViewer. The following skins are included: <ul style="list-style-type: none"> • FluorescentBlue.swf • Office.swf • OliveGreen.swf • Orange.swf • VistaAero.swf • WindowsClassic.swf • XP.swf • TocPanelOptions Allow you to specify the display options of the table of contents pane. • Url Allows you to specify the relative URL of the FlashViewer control. |

- **UseClientApi** Allows you to specify whether to use the client API for the FlashViewer.
- **ViewType** Allows you to specify the page view type. Select from Single, MultiPage, or Continuous.
- **WindowMode** Allows you to specify such display options as transparency, layering, and positioning of the FlashViewer in the browser. Select from Window, Opaque and Transparent.
- **Zoom** Allows you to specify the zoom level, between 10% and 800%, at which to display the report.

HtmlExportOptions

These options apply only when you select the `HtmlViewer ViewerType`.

- **BookmarkStyle** Allows you to specify whether to use HTML bookmarks.
- **CharacterSet** Allows you to select from 15 character sets to use for the report.
- **CreateFramesetPage** Allows you to specify whether to use Frameset or Body tags in the generated HTML report.
- **IncludeHtmlHeader** Allows you to specify whether to include a header section in the generated HTML report.
- **IncludePageMargins** Allows you to specify whether to keep page margins with reports in the generated HTML.
- **MultiPage**
- **OutputType** Allows you to specify whether to use DHTML or HTML for the output.
- **RemoveVerticalSpace** Allows you to specify whether to keep white space, for example, space at the end of a page not filled with data before a page break.
- **Title** Allows you to specify the text to display in the title bar of the Web browser.

MaxReportRunTime

The maximum number of seconds that a request for a report's output waits for the report to finish executing. The default value is 10 seconds. If a report takes longer to run than the value of this property, the control makes subsequent requests at 5 second intervals for the report to see when it is finished executing.

PdfExportOptions

These options apply only when you select the `AcrobatReader ViewerType`.

- **Application** Allows you to set the value to display in the Application field in the Document Properties dialog of the Acrobat Reader.
- **Author** Allows you to set the value to display in the Author field in the Document Properties dialog of the Acrobat Reader.
- **CenterWindow** Allows you to specify whether to position the document's window in the center of the screen in the initial view when the document is opened in the Acrobat Reader.
- **ConvertMetaToPng** Allows you to specify whether to convert meta files (WMF or EMF) into PNG files in the PDF.
- **DisplayMode** Allows you to specify how to display the document: in outlines, thumbnails, full screen, or none to use the Acrobat Reader's default display mode.
- **DisplayTitle** Allows you to specify whether to display the document title from the Title property.
- **Encrypt** Allows you to specify whether to encrypt the document.
- **ExportBookmarks** Allows you to specify whether to create PDF bookmarks from any bookmarks that may be in the report.
- **FitWindow** Allows you to specify whether to resize the document's window to fit the size of the first displayed page.
- **HideMenuBar** Allows you to specify whether to hide the viewer application's menu bar when the document is active.
- **HideToolbar** Allows you to specify whether to hide the viewer application's tool bars when the document is active.
- **HideWindowUI** Allows you to specify whether to hide user interface elements in

the document's window (such as scroll bars and navigation controls), leaving only the document's contents displayed.

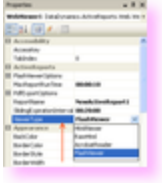
- **ImageQuality** Allows you to specify whether to render image metafiles (WMF or EMF) in the document at lowest, medium, or highest quality.
- **ImageResolution** Allows you to specify the image resolution for metafiles (WMF or EMF).
Typical values are 75-2400 dpi. 75 dpi at low resolution would be used to save space, 150 dpi is used for normal screen viewing and 300 dpi and higher is used for print quality.
- **Keywords** Allows you to set the value to display in the Keywords field in the Document Properties dialog of the Acrobat Reader. These are used in document searches.
- **NeverEmbedFonts** Allows you to specify a semicolon-delimited string of values indicating which fonts are not embedded in the PDF document. Not embedding any of the fonts used in your documents can reduce the PDF file size dramatically if you use many fonts.
- **OwnerPassword** Allows you to specify the password to enter in the reader to permit full access to the document regardless of the specified user permissions.
- **Permissions** Allows you to specify the user permissions for the document. You can combine Permissions by using commas between values.
- **Subject** Allows you to specify the value to display in the Subject field in the Document Properties dialog of the Acrobat Reader.
- **Title** Allows you to specify a title to display when the DisplayTitle property is set to True.
- **Use128Bit** Allows you to specify whether to use 128 bit encryption with full permissions capability. Set to True to enable the AllowFillIn, AllowAccessibleReaders, and AllowAssembly permissions to function. Set to False to use 40 bit encryption with limited permissions.
- **UserPassword** Allows you to specify the password to enter to allow a user to open the document in the reader. If this value is left empty, the user is not prompted for a password, but is restricted by the specified permissions.
- **Version** Allows you to specify whether to use Pdf11 (v1.1), Pdf12 (v1.2), or Pdf13 (v1.3 or Acrobat 4.0).

| | |
|----------------------------------|--|
| ReportName | This new property replaces the old Report property. The new property is a string instead of an ActiveReport object, and specifies the report to display in the viewer. |
| SkinID | Inherited from System.Web.UI.WebControls.WebControl |
| SlidingExpirationInterval | The interval in seconds between the time the report was last retrieved and the time the report is removed from the ASP.NET WebCache. |
| ViewerType | <ul style="list-style-type: none"> • HtmlViewer Provides a scrollable view of a single page of the report at a time. Downloads only HTML and javascript to the client browser. Not recommended for printable output. • RawHtml Shows all pages in the report document as one continuous HTML page. Provides a static view of the entire report document, and generally printable output, although under some circumstances pagination is not preserved. • AcrobatReader Returns output as a PDF document viewable in Acrobat Reader. Client requirements: Adobe Acrobat Reader. • FlashViewer Provides an interactive viewing experience and no-touch printing using the widely-adopted Flash Player. Client Requirements: Adobe Flash Player. |

For more information on properties inherited from the system, see the Class Library and Visual Studio help.

Web Viewer Print Options

ActiveReports provides several print options for the **WebViewer control** that you can use or modify when printing a report. The print options depend on the specific type you select in the WebViewer's **ViewerType** field.



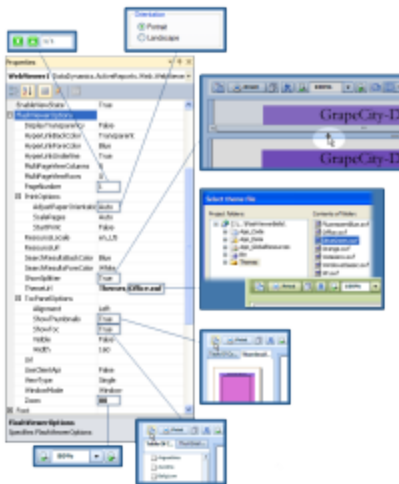
The following table describes print options details for each Web Viewer type:

| | Flash Viewer | Acrobat Reader | HTML Viewer | Raw HTML |
|---|---------------------|-----------------------|--------------------|-----------------|
| Send a report to printer (using the standard browser Print action) | + | + | + | + |
| Send a report to printer without displaying the preview dialog (one-touch printing) | + | - | - | - |
| Send a report to printer without displaying the preview and the printer dialog (no-touch printing) | - | - | - | - |
| Send a report to printer with displaying the printer and paper settings (using the viewer Print action) | + | + | - | - |
| Scale pages for printing (the ScalePages property in FlashViewerOptions/PrintOptions of the WebViewer control) | + | - | - | - |
| Adjust paper orientation for printing (the AdjustPaperOrientation property in FlashViewerOptions/PrintOptions of the WebViewer control) | + | - | - | - |

Flash Viewer Options

The new **FlashViewer** has several features which make it unique. To set up the features, select the WebViewer on your ASPX page, and in the Properties window, expand the **FlashViewerOptions** node.

See the table below for a full description of each property.



Property

Display Transparency

HyperLinkBackColor

HyperLinkForeColor

HyperLinkUnderline

MultiPageViewColumns

MultiPageViewRows

PageNumber

PrintOptions

 AdjustPaperOrientation

PrintOptions

 ScalePages

PrintOptions

 StartPrint

ResourceLocale

ResourceUrl

SearchResultsBackColor

SearchResultsForeColor

ShowSplitter

Description

Allows you to specify whether to print transparent objects.

Allows you to specify the background color of the controls that host a hyperlink.

Allows you to specify the color of the text within a control that hosts a hyperlink.

Determines whether the text in the control that hosts a hyperlink is underlined.

Determines the amount of columns to show when ViewType is MultiPage.

Determines the amount of rows to show when the ViewType is MultiPage.

Allows you to specify the page to display initially.

Select from None, Auto, or AllowScaleUp.

Select from None, Auto, or AdjustByFirstPage.

Allows you to specify whether to print the report after loading for one-touch printing. If you set the WebViewer's Height and Width properties to 0, you can have the report print without displaying the Preview dialog.

Allows you to specify the **culture** for localization. Separate multiple values with commas.

Allows you to specify the comma-separated list of URLs to SWF files with resource files.

Allows you to specify the background color of the highlighted text in the Find dialog of the WebViewer control.

Allows you to specify the color of the highlighted text in the Find dialog of the WebViewer control.

Allows you to specify whether to display the splitter, which


| | |
|-----------------------------------|--|
| ThemeUrl | allows the user to compare report pages in the viewer. Allows you to specify the relative URL of a skin to use on the FlashViewer. The following skins are included: <ul style="list-style-type: none"> • FluorescentBlue.swf • Office.swf • OliveGreen.swf • Orange.swf • VistaAero.swf • WindowsClassic.swf • XP.swf |
| TocPanelOptions Alignment | Allows you to specify the alignment of the table of contents pane. Select from Left or Right. |
| TocPanelOptions ShowThumbnails | Allows you to specify whether to display a pane with thumbnail views of pages. |
| TocPanelOptions ShowToc | Allows you to specify whether to display the table of contents in the FlashViewer. |
| TocPanelOptions Visible | Determines the visibility of the table of contents pane. |
| TocPanelOptions Width | Allows you to specify the width of the table of contents pane in pixels. |
| Url | Allows you to specify the relative URL of the FlashViewer control. |
| UseClientApi | Allows you to specify whether to use the client API for the FlashViewer. |
| ViewType | Allows you to specify the page view type. Select from Single, MultiPage, or Continuous. |
| WindowMode | Allows you to specify such display options as transparency, layering, and positioning of the FlashViewer in the browser. Select from Window, Opaque and Transparent. |
| Zoom | Allows you to specify the zoom level, between 10% and 800%, at which to display the report. |

Flash Viewer Hot Keys and Shortcuts

The following shortcuts are available on the Flash viewer:

| Keyboard Shortcut | Action |
|--------------------------|-------------------------------|
| Ctrl + F | Shows the find dialog. |
| F3 | Finds the next search result. |
| Esc | Closes the find dialog. |

| | |
|---------------------------|--|
| Page Down | Moves to the next page. |
| Page Up | Moves to the previous page. |
| Ctrl + P | Shows the print dialog. |
| Ctrl + T | Shows the table of contents. |
| Ctrl + Home | Moves to the first page. |
| Ctrl + End | Moves to the last page. |
| Ctrl + Right | Navigates forward. |
| Ctrl + Left | Navigates backward. |
| Ctrl + - | Zooms out. |
| Ctrl + + | Zooms in. |
| Left, Right, Up, Down | Moves the visible area of the page in the corresponding direction. |
| Home, End | Moves to the start or end of the current page. |
| Ctrl + o (zero) | Sets the zoom level to 100%. |
| Ctrl + rotate mouse wheel | Changes the zoom level up or down. |
| Ctrl + M | Shows multiple pages. |
| Ctrl + S | Shows a single page. |

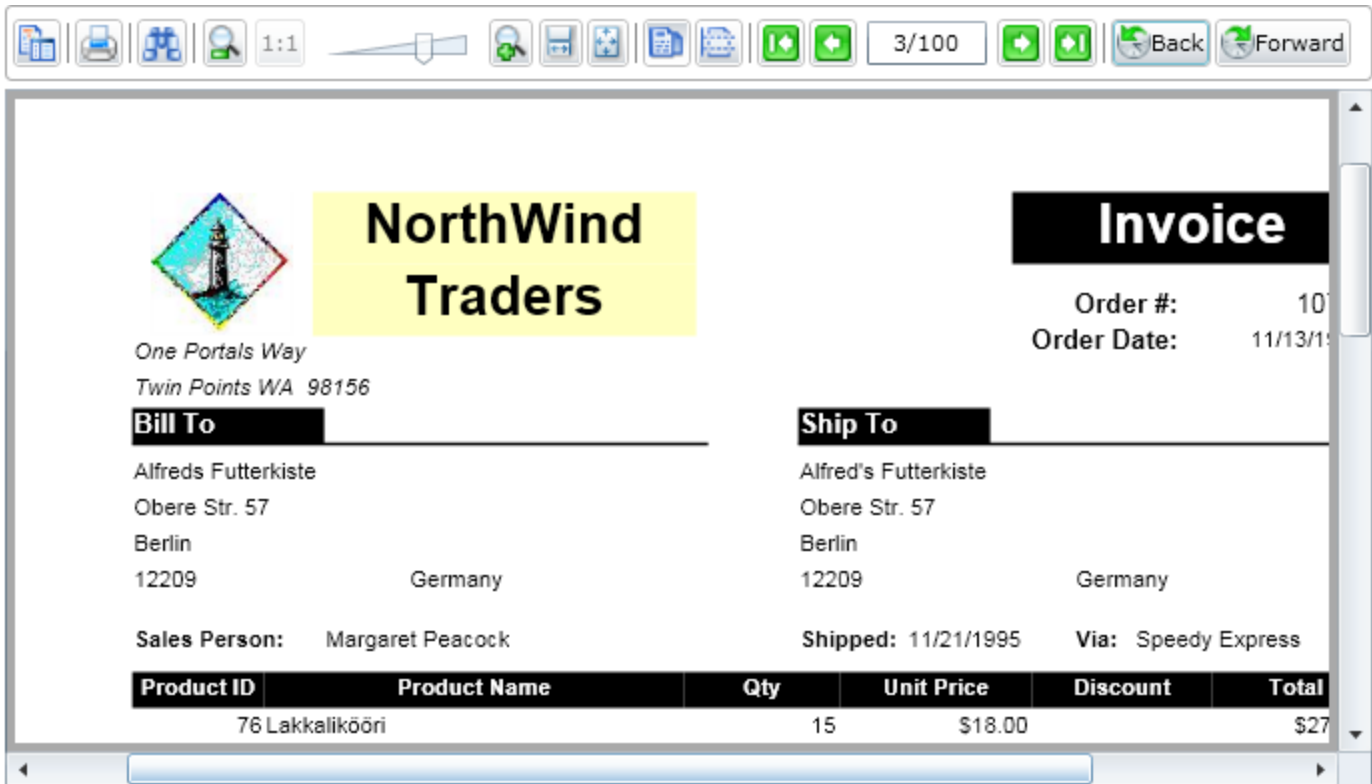
 **Note:** As with any Flash Web application, when the FlashViewer has focus, browser keyboard shortcuts do not work. The user must click outside of the FlashViewer to return focus to the browser. Likewise, if the browser has focus, the user must click inside the FlashViewer in order to use the viewer's keyboard shortcuts.

The following shortcuts are available in the Thumbnails pane of the Flash Viewer sidebar.

| Keyboard Shortcut | Action |
|--------------------------|---|
| Up | Selects the above page. |
| Down | Selects the below page. |
| Left | Selects the left page. If there is no left page, the last page of the previous row is selected. |
| Right | Selects the right page. If there is no right page, the first page of the next row is selected. |
| Page Up | Moves to the previous Thumbnail view. |
| Page Down | Moves to the next Thumbnail view. |
| Home | Selects the first page. |
| End | Selects the end page. |


Silverlight Viewer (Pro Edition)

With the new ActiveReports **Silverlight** support, you can create and preview reports in the ActiveReports **Silverlight Viewer**.









You should license your ActiveReports6 Silverlight project so that no evaluation banners appear at runtime. You can find information on licensing ActiveReports 6 Silverlight in **License Your ActiveReports** under **To license an ActiveReports6 Silverlight project**.



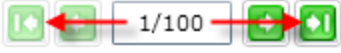

Below you can find information on the ActiveReports **Silverlight Viewer** properties and feature limitations.

 **Note:** Microsoft Silverlight 4 Tools is required for the proper application development with the ActiveReports Silverlight Viewer.

Silverlight Viewer Toolbar



| Toolbar element | Property | Description |
|---|---------------------|--|
|  | Show TOC/Thumbnails | Displays the Table of Contents and a pane with thumbnail views of pages in the Viewer. |
|  | Print | Displays the Print dialog where you can specify the printing options. The maximum value for the print range is 2000 pages. |
|  | Search | The Find option of the Silverlight Viewer. |
|  | Zoom | Specifies the Zoom level at which to display the report. |
|  | Zoom modes | Selects a mode for displaying the report. Select from Page width or Whole page. |
|  | View Type | Allows you to specify the page view type. Select from Single or Continuous. |

| | | |
|---|--------------------------|---|
|  | Go to page | Opens a specific page in a report. To view a specific page, type the page number and press ENTER. |
|  | Go to Previous/Next page | Navigates a report page by page. |
|  | Go to First/Last page | Opens the first or last page of a report. |
|  | Back/Forward | History navigation. |

Silverlight Feature Limitations

1. **Metafiles** are not supported.
2. Limitations on **printing**:
 - The ActiveReports Silverlight Viewer does not provide the **one-touch printing** option.
 - The maximum value for the print range is **2000 pages**.
 - The exceptions that occur at printing to PDF cannot be caught.
3. **Vertical text** is not supported in the ActiveReports Silverlight Viewer.
4. ActiveReports Silverlight Viewer does not support the **text strikeout** effect, **character spacing** and **text justification**.
5. Horizontal and vertical text of **MS PMincho** and some other ideographic characters may render incorrectly in the ActiveReports Silverlight Viewer.
6. The use of the Silverlight Viewer control in **layout panels** has a limitation related to the default size of the panel.
7. The **multipage mode** is not available in the ActiveReports Silverlight Viewer.
8. To display a report with **annotations** in the **Thumbnails** view correctly, hide the Thumbnails view by clicking the **Show TOC/Thumbnails** icon in the Toolbar and then open the report in the Silverlight Viewer.
9. The Silverlight **Viewer size** may get changed at runtime when the **Zoom** level is increased or decreased. This happens because the properties **HorizontalAlignment** is set to **Left** and **VerticalAlignment** is set to **Top** automatically when the Viewer control is dragged onto the Design view of MainPage.xaml. However, when both properties are set to the control's default values **Stretch**, the Viewer keeps constant despite the Zoom scale.
10. Some Silverlight **properties** and **events** are not supported in XAML. However, you can still use these properties by setting them in code as follows:

```
(viewer1 as System.Windows.Controls.Control).FontStyle = FontStyles.Italic;
```

Silverlight properties and events that are not supported

Properties


- FontStyle
- FontWeight
- Font Family
- Font Size
- Foreground
- FontStretch
- Padding
- VerticalContentAlignment
- HorizontalContentAlignment
- IsTabStop

Events


- GotFocus
- LostFocus

PDF Printing in Silverlight

You can provide PDF printing in your Silverlight project, which allows to print a document from Silverlight to the PDF format directly. This is a good alternative to the default Silverlight printing with its large print spool size issue.

 **Note:** When you print to PDF, the exceptions that occur at printing cannot be caught. Thus, if there is a printing exception, no Print dialog appears after clicking **Print** in the Silverlight Viewer toolbar.


See **Provide PDF Printing in the Silverlight Viewer** about how you can print a report from the Silverlight Viewer to the PDF format.

 **Note:** PDF printing is not supported in the Silverlight Out-of-Browser applications.

Silverlight Viewer Hot Keys and Shortcuts (Pro Edition)

The following shortcuts are available on the Silverlight Viewer:

| Keyboard Shortcut | Action |
|---------------------------|--|
| Page Down | Moves to the next page. |
| Page Up | Moves to the previous page. |
| Ctrl + Home | Moves to the first page. |
| Ctrl + End | Moves to the last page. |
| Ctrl + Right | Navigates forward. |
| Ctrl + Left | Navigates backward. |
| Ctrl + - | Zooms out. |
| Ctrl + + | Zooms in. |
| Left, Right, Up, Down | Moves the visible area of the page in the corresponding direction. |
| Home, End | Moves to the start or end of the current page. |
| Ctrl + 0 (zero) | Sets the zoom level to 100%. |
| Ctrl + rotate mouse wheel | Changes the zoom level up or down. |
| Ctrl + S | Shows a single page. |

 **Note:** Shortcuts may work differently depending on view mode – single page mode or continuous scroll mode.


The following shortcuts are available in the Thumbnails pane of the Silverlight Viewer sidebar.


| Keyboard Shortcut | Action |
|--------------------------|---|
| Up | Selects the above page. |
| Down | Selects the below page. |
| Left | Selects the left page. If there is no left page, the last page of the previous row is selected. |
| Right | Selects the right page. If there is no right page, the first page of the next row is selected. |
| Page Up | Moves to the previous Thumbnail view. |

| | |
|-----------|-----------------------------------|
| Page Down | Moves to the next Thumbnail view. |
| Home | Selects the first page. |
| End | Selects the end page. |

Medium Trust Support

ActiveReports 6 was designed for use in a Full trust environment where all its features are available without restrictions. You can use ActiveReports 6 under Medium trust, but with limitations on some of the features.


 **Caution:** Assemblies placed in the Global Assembly Cache, or GAC (C:\WINDOWS\ASSEMBLY), have Full trust permissions, so the results on your deployment machine may differ from those on your development machine.

 **Note:** If you see an evaluation banner when deploying your ActiveReports 6 project, you should use the **Web Key Generator** utility to create the Web Key and integrate the license information into your project.

Feature Limitations

1. Exporting

- **Rtf, Text, Tiff** and **Excel** filters are not supported in Medium trust.
- For the **Pdf** export filter, **digital signatures** are not supported.
- The **Pdf** export filter cannot access **System fonts**, so you must create a **Custom Font Factory (Pro Edition)** to embed any necessary fonts for non-ASCII characters.

 **Note:** If justified text and some character sets are not exported correctly, ensure that the **Custom Font Factory (Pro Edition)** is set up correctly.

2. The **End User Designer** and **Windows Form Viewer** controls require Full trust.
3. The **Picture** control does not support **metafiles**, which require Full trust.
4. The **ImageType** property of the **Chart** must be set to **PNG**.
5. **OleObject** and **Custom** controls require Full trust.
6. **Scripting** requires Full trust, so if you need to use code in reports under Medium trust, use code-based reports rather than RPX format.

Recommended Development Environment for Medium Trust Tests

To set up a Medium trust environment

Paste the following code between the <system.web> and </system.web> tags.

XML code. Paste BETWEEN the system.web tags.

```
<trust level="Medium"></trust>
```


To set up the PrintingPermission level

Most hosting providers disable the printing permissions in a partially trusted environment.

1. Open the web_mediumtrust.config file (located in the \Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\ folder).
2. Set the PrintingPermission level to NoPrinting.


XML code. Paste BETWEEN the system.web tags.

```
<IPermission class="PrintingPermission"version="1"Level="NoPrinting"/>
```

 **Note:** The default set of medium trust permissions is available in the `web_mediumtrust.config.default` file (located in the `\Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\` folder).

Windows Azure Support

You can use ActiveReports 6 on Windows Azure in full trust or partial trust modes, but with limitations on some of the features.

 **Note:** If you see an evaluation banner when deploying your ActiveReports 6 project, you should use the **Web Key Generator** utility to create the Web Key and integrate the license information into your project.

Full Trust Feature Limitations

Exporting

- **Rtf** and **Text** filters are not supported on Windows Azure.

Partial Trust Feature Limitations

1. **Exporting** limitations
 - **Rtf**, **Text**, **Tiff** and **Excel** filters are not supported in Azure partial trust.
 - **Digital signatures** for the **Pdf** export filter are not supported.
2. The **Picture** control does not support **metafiles** in Azure partial trust.
3. **Chart** and **Custom** controls with the **ImageType** property set to **metafiles** also require Full trust.
4. The **OleObject** control requires Full trust.
5. **Scripting** requires Full trust, so if you need to use code in reports under partial trust, use code-based reports rather than RPX format.
6. The **OLE DB data source** is not supported in Azure partial trust.

Recommended Development Environment for Azure Partial Trust

To set up Partial trust in an Azure Web Role Application

1. In Solution Explorer, expand the Roles folder of your Azure project and double-click the role reference **WebRole1**.
2. On the **Configuration** page that appears, select **Windows Azure partial trust** under .Net trust level.
3. In Solution Explorer under WebRole1, double-click Web.config.
4. In the Web.config file that opens, remove the `<System.Diagnostics>` tag.
5. In Solution Explorer under WebRole1, double-click WebRole.cs.
6. In the WebRole.cs file that opens, remove or comment the following line as the diagnostics library does not support partial trust callers:

```
DiagnosticMonitor.Start("DiagnosticsConnectionString")
```

To set up Partial trust in an Azure Worker Role Application

1. In Solution Explorer, expand the Roles folder of your Azure project and double-click the role reference **WorkerRole1**.

2. On the **Configuration** page that appears, select **Windows Azure partial trust** under .Net trust level.
3. In Solution Explorer under WorkerRole1, double-click App.config.
4. In the App.config file that opens, remove the <System.Diagnostics> tag.
5. In Solution Explorer under WorkerRole1, double-click WorkerRole.cs.
6. In the WorkerRole.cs file that opens, remove or comment the following line as the diagnostics library does not support partial trust callers:

```
DiagnosticMonitor.Start("DiagnosticsConnectionString")
```

To set up a local storage resource in a Web Role partial trust Application

1. Open the service definition file (ServiceDefinition.csdef) and add the following tag after <WebRole name="WebRole1">:

Paste into ServiceDefinition.csdef after <WebRole name="WebRole1">


```
<LocalStorage name="WebRoleStorage" cleanOnRoleRecycle="true" />
```

2. To export to PDF, paste the following code into Default.aspx.cs to the button_Click event:

Paste into Default.aspx.cs to the button_Click event

```
NewActiveReport1 rpt=new NewActiveReport1 ();
rpt.Run ();
DataDynamics.ActiveReports.Export.Pdf.PdfExport pdfe = new
DataDynamics.ActiveReports.Export.Pdf.PdfExport ();
string webrolestorageName = "WebRoleStorage";
string webRoleFileName = "Test.pdf";
LocalResource lr = RoleEnvironment.GetLocalResource (webrolestorageName);
string path = lr.RootPath + webRoleFileName;
pdfe.Export (rpt.Document, path);
```

3. You can find the exported PDF file at the following location:
C:\Users\stduser\AppData\Local\dfmp\so\deployment (o)\res\deployment (o).CloudService1.WebRole1.o\directory.

 **Note:** The file is available only at the application runtime.

To set up a local storage resource in a Worker Role partial trust Application

1. Open the service definition file (ServiceDefinition.csdef) and add the following tag after <WorkerRole name="WorkerRole1">:

Paste into ServiceDefinition.csdef after <WorkerRole name="WorkerRole1">


```
<LocalStorage name="WorkerRoleStorage" cleanOnRoleRecycle="true" />
```

2. To export to PDF, add the following code into Default.aspx.cs to the button_Click event:

Paste into Default.aspx.cs to the button_Click event

```
NewActiveReport1 rpt=new NewActiveReport1 ();
rpt.Run ();
DataDynamics.ActiveReports.Export.Pdf.PdfExport pdfe = new
DataDynamics.ActiveReports.Export.Pdf.PdfExport ();
string workerrolestorageName = "WorkerRoleStorage";
string workerRoleFileName = "Test.pdf";
LocalResource lr = RoleEnvironment.GetLocalResource (workerrolestorageName);
string path = lr.RootPath + workerRoleFileName;
pdfe.Export (rpt.document, path);
```

3. You can find the exported PDF file at the following location:
C:\Users\stduser\AppData\Local\dftmp\so\deployment (o)\res\deployment (o).CloudService1.WorkerRole1.o\directory.

 **Note:** The file is available only at the application runtime.

Concepts

This section introduces you to the basic structure and concepts behind ActiveReports 6 to enable you to efficiently create reports.

This section contains information about:

Report Structure

ActiveReports are arranged in bands similar to those found in Access reports. This topic describes each type of section and how it may be used.

Report and Page Settings

Find out how to access printer setup, page settings, styles, and global report settings at design time or run time.

KeepTogether Options

There are several ways that you can work with ActiveReports to keep section data together. Learn about all of them here.

Date, Time, and Number Formatting

Learn about built-in and customizable formatting for date, time, currency, and other numeric values.

BarCodes

Learn to format the many barcode styles included with the BarCode control.

Parameters

ActiveReports allows you to specify parameters for simple reports and subreports, in SQL statements, or at run time. This topic explains the various methods.

Layout Files

Report layouts can be saved to different file formats. Find out which one is best for you.

Scripting

In order to save custom code along with an rpx report layout file, you must use scripts. Here you will find a number of important concepts about scripting.

Export Filters

ActiveReports export filters can do a lot, but each format has inherent limitations. This topic explains them so that you can avoid the frustration of trying to do the impossible.

Charts

The ActiveReports chart control allows you to create many types of charts, including XY and financial charts. In this section, you can find information on each of these chart types, as well as on different ways to add data, and on customizing the appearance of charts.

RichText

Learn how the RichText control works, and which HTML tags are supported.

Grouping Data

Find out how grouping works in ActiveReports, and what special options help you to control groupings.

Multiple Groupings

Learn about nesting groups, and how to manage multiple groups.

Subreports

Find out when and how to use subreports.

Report Events

Learn about the intelligent, multi-threaded, single-pass processing used in ActiveReports.

Section Events

Each section in an ActiveReport has three events. Learn what you can and cannot do within these events.

Sequence of Events

Sometimes the order in which report events fire can affect the way in which your reports run. This topic helps you to understand what determines the sequence in which they fire.

Unbound Reporting

Learn which events you can use to set up unbound reporting, and copy some code samples to get started quickly.

Optimizing ActiveReports

If you run very large reports, there are several ways that you can make them run more quickly and use less memory.

CacheToDisk and Resource Storage

If you run very large reports, and are considering whether to use CacheToDisk to use less memory, here is insight into what goes on behind the scenes to help you with your decision.

Section 508 Compliance

Learn about accessibility support in ActiveReports components.

Localization

Find out how you can localize the viewer control and other ActiveReports controls.

Designer Control (Pro Edition)

The designer control allows you to offer your end users the features you enjoy with ActiveReports, customized to your specifications.

Report Structure

ActiveReports are based on banded sections. By default, an ActiveReport has three sections: a PageHeader, a Detail section, and a PageFooter. You can right-click on the report and select Insert, then select the type of section pair you would like to add: ReportHeader and Footer, or GroupHeader and Footer. Except for the Detail section, sections always come in pairs.

A report section contains a group of controls that are processed and printed at the same time as a single unit. All sections except the detail section come in pairs, above and below the detail section. You can hide any section that you are not using by setting the Visible property of the section to False. 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 only needs to appear once at the report's start. This section has a NewPage property that you can use to cause the report to break to a new page after it renders.

Page Header

A report can have one page header section that prints at the top of each page. Unless the page contains a report header section, the page header is the first section that prints on the page. 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.

GroupHeader

A report can consist of single or nested groups, with each group having its own header and footer sections. The header section is inserted and printed immediately before the detail section. For more information on grouping, see **Grouping Data**.

The GroupHeader section is the only section type on which you can drop the new CrossSectionBox and CrossSectionLine controls, which then span any intervening sections to the corresponding GroupFooter section.

For **Columnar Reports**, you can have the GroupHeader section follow the ColumnLayout or not, use ColumnGroupKeepTogether, and select whether to start a NewColumn before or after a group.

You can also specify whether to print a NewPage before or after the section, and have the section print on every page until the group details complete with the RepeatStyle property. The UnderlayNext property allows you to show group header information inside the group details, so long as you keep the BackColor property of the Detail section set to Transparent.

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. You can set the CanShrink property to True to eliminate white space after controls, and you can set up **Columnar Reports** using ColumnCount, ColumnDirection, ColumnSpacing and NewColumn properties.

The KeepTogether property attempts to keep the section together on a single page, and the new **RepeatToFill** property allows you to fill each page with the same number of formatted rows, regardless of whether there is enough data to fill them. This is especially useful for reports such as invoices in which you want consistent formatting like lines or green bars or back colors to fill each page down to the Footer section at the bottom.



Note: The **RepeatToFill** property cannot be used if the PageBreak or SubReport control is used in the Detail section, or if the **NewPage** or **NewColumn** property is set to any value other than **None**.

When you use this property when two groups are present, the second GroupFooter section is always on the next page. It processes correctly only with a single grouping. The ReportFooter section also prints on the next page when you set this property to **True**.

GroupFooter

A report can consist of single or 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.

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.

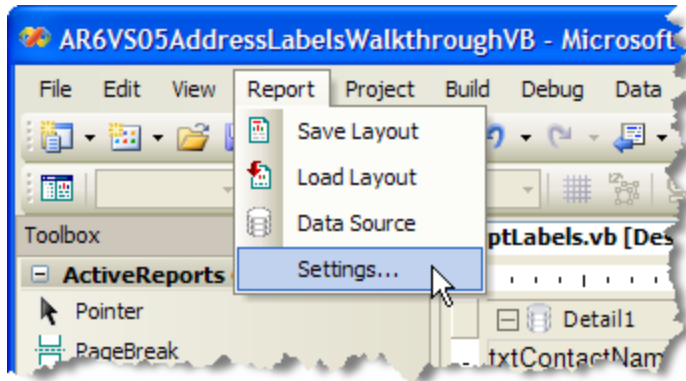
Report Footer

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

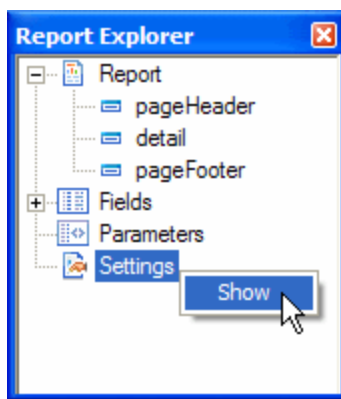
Report and Page Settings

The Report Settings Dialog

With ActiveReports, you can modify facets of your report, such as the page setup, printer settings, styles, and global settings at design time, as well as at run time. To make changes at design time, access the Report Settings dialog by selecting **Report**, then **Settings** from the toolbar menu.



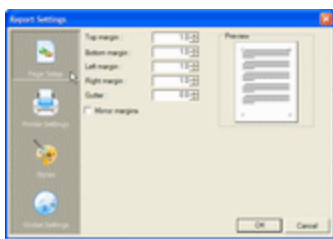
Alternatively, you can right-click the **Settings** node in the Report Explorer and select **Show**.



Page Setup

On the Page Setup page, you can make changes to the report margins (left, right, top, and bottom), specify a gutter, and select the Mirror margins option.

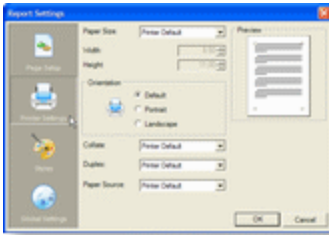
By setting a gutter and selecting Mirror margins, you can easily set up reports for publishing purposes. When you select Mirror margins, the inner margins in the report are set for opposite pages to be the same width and the outside margins for opposite pages to be the same width. Specifying a gutter gives extra space between the edge of the page and the margins. This allows reports to be bound.



Printer Settings

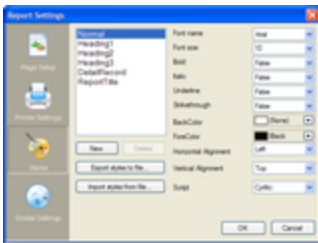
On the Printer Settings page, you can make changes to the printer paper size and orientation. You can set a custom paper size by dropping down the **Paper Size** list and selecting **Custom Size**. Once you select this option, the **Width** and **Height** fields are enabled.

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



Styles

On the Styles page, you can change the appearance of text associated with controls, either by creating a new style sheet, or by modifying and applying an existing style. See **Use External Style Sheets** for more information.

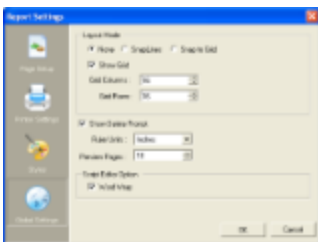


Global Settings

On the Global Settings page, you can change the design layout of your report. You can use **SnapLines**, show or hide the grid, set the controls to align to the grid, have a warning appear when you try to delete a parameter or calculated field from the Report Explorer, set the number of columns or rows on the grid, and change the ruler units to inches or centimeters.

Also, you can set the number of Pages that display in the Preview tab of the ActiveReports Designer. The minimum value is **1** and the maximum value is **10000**. By default, the Preview tab displays 10 pages. This is generally enough to allow you to see all of your report sections without taking the time to generate the entire report.

By setting the **WordWrap** option in the Script Editor, you allow to automatically wrap lines that exceed the Editor's window size to the next line. If this option is not set, the Script Editor displays the horizontal scrollbar that you can use to view elements that are outside the Editor's viewing area.



KeepTogether Options

ActiveReports provides several KeepTogether options for your reports so that you can keep sections together when you print them.

- **KeepTogether** is available on the following sections:
 - GroupHeader
 - Detail

- GroupFooter
- ReportFooter
- The GroupHeader section has two additional keep together options:
 - **GroupKeepTogether**
 - **ColumnGroupKeepTogether**

KeepTogether

The KeepTogether property, when set to True, attempts to print the section on a single page with no page breaks. If the section is too large for the current page and too large to fit fully on the next page, the KeepTogether property is ignored.

Setting the property to False allows the section to split across two or more pages.

GroupKeepTogether

The GroupKeepTogether property, which can be set on a group header section, has three enumerated values:

- **None** is the default setting. The group header does not attempt to stay with its related sections, so the group block is allowed to split across pages.
- **FirstDetail** keeps the group header and at least the first detail together on the same page to prevent widowed group header sections. If there is no room on the current page for the first detail, the group header moves to the next page along with the detail.
- **All** attempts to keep the section, related details, and group footer as a single block on the same page. If the group block does not fit on a single page, the property is ignored.

For more information on grouping, see the **Grouping Data** topic.

ColumnGroupKeepTogether

The ColumnGroupKeepTogether property only takes effect when the GroupHeader's GroupKeepTogether property is set to All, and the Detail section's Columns property is set to a value greater than 1. When set to true, it keeps newspaper-style column layouts in both the Detail and Group sections together. It attempts to prevent a group block from splitting across columns. If a group cannot fit in the current column, it tries the next. If the group is too large for a single column, the property is ignored.

Setting the property to False allows the group block to split across two or more columns. For more information on columnar reports, see the **Columnar Reports** walkthrough.

Date, Time, and Number Formatting

ActiveReports allows you to set formatting strings for date, time, currency, and other numeric values using the OutputFormat property on the TextBox control. The OutputFormat dialog also allows you to select international currency values and select from various built-in string expressions. In addition to the built-in string expressions, you may use any .NET standard formatting strings. You can find information about these strings ([Numerics](#) and [Date/Time](#) formats) on MSDN.



Note: The **ReportInfo** control has many preformatted options for RunDateTime and Page Numbers. For more information, see **ActiveReports Toolbox Controls**.

The OutputFormat property allows four sections delimited by a semicolon. Each section contains the format specifications for a different type of number:

- The first section provides the format for positive numbers.
- The second section provides the format for negative numbers.
- The third section provides the format for Zero values.

- The fourth section provides the format for Null or System.DBNull values.

Below is the example of how to set the OutputFormat property.

To write code in Visual Basic.NET

Visual Basic.NET code

```
TextBox1.OutputFormat = " $#, #00.00; ($#, #00.00); $0.00; ""#"""
```

To write code in C#

C# code

```
TextBox1.OutputFormat = " $#, #00.00; ($#, #00.00); $0.00; \"#\";"
```

Underscore support

You can use the underscore character in the OutputFormat property to add the white space after the character specified in the **Value** property. For example, using the underscore character adds the white space for a closing parenthesis in a positive number format while the negative number format includes parentheses. This way both positive and negative values are lined up at the decimal point.

| |
|--------|
| 1.23 |
| (1.23) |

Below is the example of how to set the OutputFormat property with the underscore character.

To write code in Visual Basic.NET

Visual Basic.NET code

```
TextBox1.OutputFormat = "0.00_); (0.00) "  
TextBox1.Value = 1.23  
TextBox2.OutputFormat = "0.00_); (0.00) "  
TextBox2.Value = -1.23
```

To write code in C#

C# code

```
TextBox1.OutputFormat = "0.00_); (0.00) ";  
TextBox1.Value = 1.23;  
TextBox2.OutputFormat = "0.00_); (0.00) ";  
TextBox2.Value = -1.23;
```

Using the underscore character in the **OutputFormat** property does not take effect in the combination with the following properties.

- The **RightToLeft** property is set to **True**.
- The **VerticalText** property is set to **True**.
- The **Alignment** property is set to **Justify**.
- The **TextJustify** property is set to **DistributeAllLines**.

Times:

- hh:mm tt = 09:00 AM
- HH:mm = 21:00 (twenty-four hour clock)
- HH = hours in 24 hour clock
- hh = hours in 12 hour clock
- mm = minutes
- ss = seconds
- tt = AM or PM

Dates:

- dddd, MMMM d, yyyy = Saturday, December 25, 2004
- dd/MM/yyyy = 25/12/2004
- d or dd = day in number format
- ddd = day in short string format (for example, Sat for Saturday)
- dddd = day in string format (for example, Saturday)
- MM = month in number format
- MMM = month in short string format (for example, Dec for December)
- MMMM = month in string format (for example, December)
- y or yy = year in two digit format (for example, 04 for 2004)
- yyyy or YYYY = year in four digit format (for example, 2004)

Currency and numbers:

- \$0.00 = \$6.25
- \$#, #00.00 = \$06.25
- 0 = digit or zero
- # = digit or nothing
- % = percent-multiplies the string expression by 100

BarCodes

The ActiveReports BarCode control offers all of the following barcode styles:

Barcode styles and descriptions



Note: The **RSS** and **QRCode** styles have fixed height-to-width ratios. When you resize the width, the height is automatically calculated.

| BarCodeStyle | Description |
|--------------|---|
| Ansi39 | ANSI 3 of 9 (Code 39) uses upper case, numbers, -, * \$ / + %. This is the default barcode style. |
| Ansi39x | ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set. |
| Code_2_of_5 | Code 2 of 5 uses only numbers. |
| Code25intlv | Interleaved 2 of 5 uses only numbers. |
| Code25mat | Code 25 Matrix is a two-dimensional version of the linear Code 2 of 5 barcode. |
| Code39 | Code 39 uses numbers, % * \$ / . , - +, and upper case. |
| Code39x | Extended Code 39 uses the complete ASCII character set. |

| | |
|----------------|---|
| Code_128_A | Code 128 A uses control characters, numbers, punctuation, and upper case. |
| Code_128_B | Code 128 B uses punctuation, numbers, upper case and lower case. |
| Code_128_C | Code 128 C uses only numbers. |
| Code_128auto | Code 128 Auto uses the complete ASCII character set. Automatically selects between Code 128 A, B and C to give the smallest barcode. |
| Code_93 | Code 93 uses uppercase, % \$ * / , + -, and numbers. |
| Code93x | Extended Code 93 uses the complete ASCII character set. |
| MSI | MSI Code uses only numbers. |
| PostNet | PostNet uses only numbers with a check digit. |
| Codabar | Codabar uses A B C D + - : . / \$ and numbers. |
| EAN_8 | EAN-8 uses only numbers (7 numbers and a check digit). |
| EAN_13 | EAN-13 uses only numbers (12 numbers and a check digit). If there are only 12 numbers in the string, it calculates a checksum and adds it to the thirteenth position. If there are 13, it validates the checksum and throws an error if it is incorrect. |
| UPC_A | UPC-A uses only numbers (11 numbers and a check digit). |
| UPC_Eo | UPC-Eo uses only numbers. Used for zero-compression UPC symbols. For the Caption property, you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must be zero) UPC-A code. If an 11-digit code is entered, the Barcode control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed. |
| UPC_E1 | UPC-E1 uses only numbers. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters. |
| RM4SCC | Royal Mail RM4SCC uses only letters and numbers (with a check digit). This is the barcode used by the Royal Mail in the United Kingdom. |
| UCCEAN128 | UCC/EAN –128 uses the complete ASCII character Set. This is a special version of Code 128 used in HIBC applications. |
| QRCode | QRCode is a 2D symbology that is capable of handling numeric, alphanumeric and byte data as well as Japanese kanji and kana characters. This symbology can encode up to 7,366 characters. |
| Code49 | Code 49 is a 2D high-density stacked barcode containing two to eight rows of eight characters each. Each row has a start code and a stop code. Encodes the complete ASCII character set. |
| JapanesePostal | This is the barcode used by the Japanese Postal system. Encodes alpha and numeric characters consisting of 18 digits including a 7-digit postal code number, optionally followed by block and house number information. The data to be encoded can include hyphens. |
| Pdf417 | Pdf417 is a popular high-density 2-dimensional symbology that encodes up to 1108 bytes of information. This barcode consists of a stacked set of smaller barcodes. Encodes the full ASCII character set. It has ten error correction levels and three data compaction modes: Text, Byte, and Numeric. This symbology can encode up to 1,850 alphanumeric characters or 2,710 numeric characters. |
| EAN128FNC1 | EAN-128 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry. This type of bar code contains the following sections: |

- Leading quiet zone (blank area)
- Code 128 start character
 - A allows standard alphanumeric plus control and special characters
 - B allows standard alphanumeric plus lower case alpha and special characters
 - C allows a set of 100 digit pairs from 00 to 99
- FNC (function) 1 character which allows scanners to identify this as an EAN-128 barcode
- Data (AI plus data field)
- Symbol check character (Start code value plus product of each character position plus value of each character divided by 103. The checksum is the remainder value.)
- Stop character
- Trailing quiet zone (blank area)

The AI in the Data section sets the type of the data to follow (i.e. ID, dates, quantity, measurements, etc.). There is a specific data structure for each type of data. This AI is what distinguishes the EAN-128 code from Code 128.

Multiple AIs (along with their data) can be combined into a single bar code.

EAN128FNC1 is a UCC/EAN-128 (EAN128) type barcode that allows you to insert FNC1 character at any place and adjust the bar size, etc., which is not available in UCC/EAN-128.

To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.

| | |
|-----------------------------|--|
| RSS14 | RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning. |
| RSS14Truncated | RSS14Truncated uses the EAN.UCC information as in the RSS14, but also includes Indicator digits of zero or one for use on small items not scanned at the point of sale. |
| RSS14Stacked | RSS14Stacked uses the EAN.UCC information with Indicator digits as in the RSS14Truncated, but stacked in two rows for a smaller width. |
| RSS14StackedOmnidirectional | RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width. |
| RSSExpanded | <p>RSSExpanded uses the EAN.UCC information as in the RSS14, but also adds AI elements such as weight and best-before dates.</p> <p>RSSExpanded allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs).</p> <p>To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.</p> |
| RSSExpandedStacked | <p>RSSExpandedStacked uses the EAN.UCC information with AI elements as in the RSSExpanded, but stacked in two rows for a smaller width.</p> <p>RSSExpandedStacked allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs).</p> <p>To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.</p> |

The following properties help you to customize the exact barcode you need for your application:

Barcode properties and descriptions

| Property | Description |
|-----------------|--|
| Alignment | The horizontal alignment of the caption in the control. Select from Near, Center, or Far. See CaptionPosition for vertical alignment. |
| AutoSize | When set to True, the barcode automatically stretches to fit the control. |
| BackColor | Select a background fill color for the barcode. |
| BarWidth | Set the width, in inches, of the barcode's narrow bars. Setting the width to 0 expands the barcode to fit the control. The width ratio is 1 to 0.012 inches. So setting the BarWidth to 2 will have a value of 0.024 inches, while a value of 10 yields a bar width of 0.12 inches for the narrowest bars. |
| CaptionPosition | The vertical alignment of the caption in the control. Select from None, Above, or Below. See Alignment for horizontal alignment. None is selected by default, and no caption is displayed. |
| ChecksumEnabled | Some barcode styles require a checksum and some have an optional checksum. CheckSumEnabled has no effect if the style already requires a check digit or if the style does not offer a checksum option. |
| Code128 | Code128 has three settings that work in conjunction: Dpi, BarAdjust, and ModuleSize. This property only applies to the barcode style EANFNC1. You can improve the readability of the barcode by setting all three properties. <ul style="list-style-type: none"> • Dpi sets the printer resolution. Specify the resolution of the printer as dots per inch to create an optimized barcode image with the specified Dpi value. • BarAdjust sets the adjustment size by dot units, which affects the size of the module and not the entire barcode. • ModuleSize sets the horizontal size of the barcode module. |
| Code49 | Code49 Options include Grouping and Group. If Grouping is set to True, any value not expressed by a single barcode is expressed by splitting it into several barcodes, and the Group property may be set to a number between 0 and 8. The default values are False and 0, respectively. When the Group property is set to 2, the grouped barcode's second symbol is created. When invalid group numbers are set, the BarCodeDataException is thrown. |
| Direction | Specify the print direction of the barcode symbol. Select from LeftToRight (the default value), RightToLeft, TopToBottom, or BottomToTop. |
| Font | Set the font for the caption. Only takes effect if you set the CaptionPosition property to a value other than None. |
| ForeColor | Select a color for the barcode and caption. |
| PDF417 | PDF417 Options only apply to the barcode style PDF417. <ul style="list-style-type: none"> • Column sets column numbers for the barcode. Values for this property range from 1 to 30. The default value is -1 which automatically determines row numbers. • ErrorLevel sets the error correction level for the barcode. Values range between 0 and 8. The error correction capability increases as the value increases. With each increase in the ErrorLevel value, the size of the barcode increases. The default value is -1 for automatic configuration. • Row sets row numbers for the barcode. Values range between 3 and 90. The default value is -1 which automatically determine row numbers. • Type sets the barcode type to Normal or Simple. Simple is the compact type in which the right indicator is neither displayed nor printed. |
| QRCode | QRCode Options only apply to the barcode style QRCode. <ul style="list-style-type: none"> • Connection allows any value which cannot be expressed by a single barcode to split into several barcodes. This property is used in conjunction with the ConnectionNumber property. • ConnectionNumber Use this property with the Connection property to set the number of barcodes it can split into. Values between 0 and 15 are valid. An invalid number raises the BarCodeData Exception. |

- ErrorLevel values are L (7% restorable), M (15% restorable), Q (25% restorable), and H (30% restorable). The higher the percentage, the larger the barcode becomes.
- Mask is used to balance brightness and offers 8 patterns in the QRCodeMask enumeration. The default value is Auto, which sets the masking pattern automatically, and is recommended for most uses.
 - Mask000 $(i+j) \bmod 2 = 0$
 - Mask001 $i \bmod 2 = 0$
 - Mask010 $j \bmod 3 = 0$
 - Mask011 $(i+j) \bmod 3 = 0$
 - Mask100 $((i \div 2) + (j \div 3)) \bmod 2 = 0$
 - Mask101 $(ij) \bmod 2 + (ij) \bmod 3 = 0$
 - Mask110 $((ij) \bmod 2 + (ij) \bmod 3) \bmod 2 = 0$
 - Mask111 $((ij) \bmod 3 + (i+j) \bmod 2) \bmod 2 = 0$
- Model sets Model1, the original model, or Model2, the extended model.
- Version indicates the size of the barcode. As the value increases, the barcode's size increases, enabling more information to be stored. Specify any value between 1 and 14 when the Model property is set to Model1 and 1 to 40 for Model2. The default value is -1, which automatically determines the version most suited to the value.

| | |
|----------|---|
| RowCount | Sets the number of stacked rows in the barcode. |
| Style | Sets the symbology used to render the barcode. See the table above for details about each style. |
| Text | Sets the value to print as a barcode symbol and caption. ActiveReports fills this value from the bound data field if the control is bound to the data source. |

Barcode Limitations

- Some barcode types may render incorrectly and contain white lines in the Html and RawHtml views. However, this limitation does not affect printing and scanning.

The list of barcode types that may render with white lines in the Html and RawHtml views

- Code49
- QRCode
- Pdf417
- RSSExpandedStacked
- RSS14Stacked
- RSS14StackedOmnidirectional

Parameters

You can use the ActiveReports Parameters collection to pass values directly into a textbox or a chart on a report, or to choose what subset of data from your data source to display in a particular instance of a report, or to pass values from a main report into a subreport. There are several ways in which you can collect values for parameters:

- You can prompt the user for parameter values.
- You can get parameter values from the main report and pass them into a subreport.
- You can collect parameter values from a control in a Web form or a Windows form.

There are also several ways in which you can set up parameters for a report:

- You can enter syntax like the following into your SQL query: `<%Name | PromptString | DefaultValue | DataType | PromptUser%>`
- You can add parameters to the Report Explorer.

- You can add parameters to the code behind the report, in the ReportStart event.

Collecting Parameter Values

In order to prompt the user for parameter values, all of the following must be in place:

- At least one parameter exists in the Parameters collection of the report.
- The **PromptUser** property for at least one parameter is set to **True**.
- On the report object, the **ShowParameterUI** property must be set to **True**.

When there are parameters in the collection and the ShowParameterUI property is True, the user prompt automatically displays when the report is run. When the user enters the requested values and clicks the OK button, the report displays using the specified values.

Values of a parameter added to the Report Explorer can be applied to a parameter in the SQL query - you should just specify the "param:" prefix for a parameter in the SQL query. Specifying the "param:" prefix for a parameter in the SQL query relates this parameter to the one in the Report Explorer. For example, "select * from customers where customername = '<%param:Parameter1%>'". In this case, the parameter with the "param:" prefix in the SQL query is updated with values of the corresponding parameter in the Report Explorer.



Tip: Within the same report, you can prompt users for some parameters and not for others by setting the **PromptUser** property to **True** on some and **False** on others.

However, if the report object's **ShowParameterUI** property is set to **False**, the user prompt does not display for any parameters regardless of its **PromptUser** setting.

In order to collect parameters from a main report to pass into a subreport, all of the following must be in place:

- The SQL queries for both reports must contain the same field.
- The subreport's **ShowParameterUI** property must be set to **False**.
- The subreport's SQL query must contain the parameter syntax with the Name value set to the name of the field that is common to both reports.

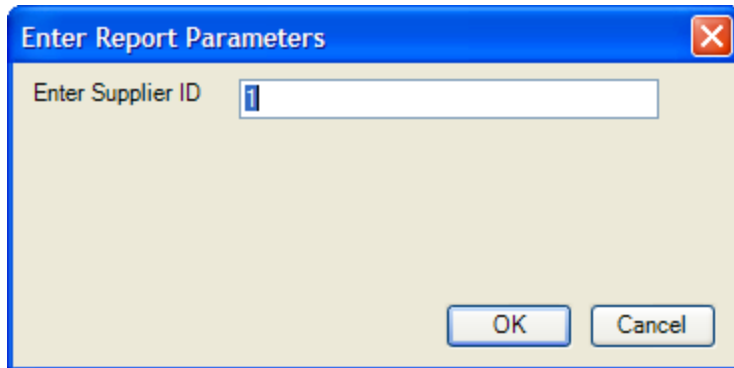
To collect parameter values from a Windows form or a Web form, use code to collect the values into variables, and then pass them into the report's ReportStart event. See sample code in the **Add Parameters** topic. In this case, the report's **ShowParameterUI** property must be set to **False**.

Adding Parameters to the Parameters Collection via the SQL Query

When you add a single parameter to a report's Parameters collection via the SQL query, a query that looks like the following creates a user prompt like the one pictured below.

SQL Query.


```
SELECT * FROM Products INNER JOIN Categories ON Products.CategoryID =  
Categories.CategoryID  
WHERE Products.SupplierID = <%SupplierID|Enter a Supplier ID|1|S|True%>
```



There are five values in the parameter syntax, separated by the pipe character: |

Only the first value (Name) is required, but if you do not specify the third value (DefaultValue), the field list is not populated at design time. You can provide only the Name value and no pipes, or if you wish to provide some, but not all of the values, simply provide pipes with no space between them for the missing values. For example, `<%ProductID||||False%>`

- **Name** This is the unique name of the parameter, and corresponds to the Key property in parameters entered via code.
- **PromptString** This string is displayed in the user prompt to let the user know what sort of value to enter.
- **DefaultValue** Providing a default value to use for the parameter allows ActiveReports to populate the bound fields list while you are designing your report, enabling you to drag fields onto the report. It also populates the user prompt so that the user can simply click the OK button to accept the default value.
- **Type** This value, which defaults to S for string, tells ActiveReports what type of data the parameter represents. It also dictates the type of control used in the user prompt. The type can be one of three values.
 - **S** (string) provides a textbox into which the user can enter the string.


 **Note:** Depending on your data source, you may need to put apostrophes (single quotes) or quotation marks around the parameter syntax for string values.

For example, `'<%MyStringParameter%>'`

Also, if you provide a default value for a string parameter that is enclosed in apostrophes or quotation marks, ActiveReports sends the apostrophes or quotation marks along with the string to SQL.


For example, `<%MyStringParameter||"DefaultValue"|S|False%>`

- **D** (date) provides a drop-down calendar control from which the user can select a date.

 **Note:** Depending on your data source, you may need to put number signs around the parameter syntax.

For example, `#<%MyDateParameter%>#`

B (Boolean) provides a checkbox which the user can select or clear.

 **Note:** If you provide a default value of True or False, or 0 or 1 for a Boolean parameter, ActiveReports sends it to SQL in that format.


- **PromptUser** This Boolean allows you to tell ActiveReports whether to prompt the user for a value. This can be set to **True** for some parameters and **False** for others. If you set the report's **ShowParameterUI** property to **False**, users are not prompted for any parameters, regardless of the **PromptUser** value set for any parameter in the report.

For a date parameter, you can use a SQL query like the following to allow users to select a beginning and ending date.

SQL Query.

```
SELECT * FROM Orders INNER JOIN [Order Details] ON Orders.OrderID = [Order
Details].OrderID
```

```
WHERE OrderDate BETWEEN #<%StartDate|Start date|1/1/1994|D|True%># AND
#<%EndDate|End date|12/31/1994|D|True%>#
```

 **Note:** Specifying the "param:" prefix for a parameter in the SQL query relates this parameter to the one in the Report Explorer.

For example, "select * from customers where customername = '<%param:Parameter1%>'".

In this case, the parameter with the "param:" prefix in the SQL query is updated with values of the corresponding parameter in the Report Explorer.

Layout Files

Report layouts in ActiveReports are automatically saved as C# or Visual Basic for .NET files within the project in which they are created. Each report is composed of three files:

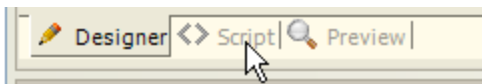
- *rptYourReportName.vb* or *.cs*
- *rptYourReportName.Designer.vb* or *.cs*
- *rptYourReportName.resx*

In this way, layout information models the behavior of Windows Forms in the .NET framework.

However, you can also save report layouts as stand-alone Report XML (RPX) files. RPX files are the same report layouts used in previous editions of ActiveReports for .NET. This makes ActiveReports 6 truly backward compatible. Older layout files can easily be brought into the newest applications while new layout files can be saved to an older format.

When you save a layout that contains a dataset, the data adapter and data connection are saved, but the dataset itself is lost. When you load the saved layout into another report, you must generate the dataset again.

The RPX format cannot contain Visual Basic.NET or C# code. In order to port logic along with the layout, you can add VB.NET or C# script in the Script view of the report.



For more information on using script with a layout file, see **Scripting**.

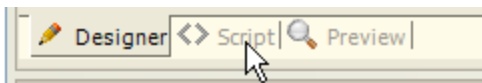
Scripting

ActiveReports allows you to use VB.NET or C# script to port your custom logic to report layouts. This permits layouts saved to report XML (RPX) files to serve as stand-alone reports. By including scripting before you save the report layout as an RPX file, you can later load, run, and display the report directly to the viewer control without using the designer. In conjunction with RPX files, scripting allows you to update distributed reports without recompiling your project.

ActiveReports loads RPX files, including any scripting, in the InitializeComponent() method.

You can add C# or VB.NET code to the script editor at design time or by using the `rpt.Script` property at run time. The script is then saved to the RPX file along with layout information.

To access the script editor, click the script tab below the report design surface.




Since the RPX file can be read with any text editor, use the **AddCode** (**AddCode Method** in the on-line documentation) or **AddNamedItem** (**AddNamedItem Method** in the on-line documentation) method to

add secure information such as a connection string.

 **Note:** The ActiveReports script editor supports IntelliSense that helps the writing of code by making the access to the language elements fast and easy.

Tips for Using Script

- **Keep the report class public.** If the report class is private, the script cannot recognize the items in your report. The report class is public by default.
- **Set the Modifiers property of any control referenced in script to Public.** If the control's **Modifiers** property is not set to **Public**, the control cannot be referenced in script and an error occurs when the report is run. The **Modifiers** property has a default value of **Private**, so you must set this property in the designer.
- **Use "this" (as in C# code-behind) or "Me" (as in VB code-behind) to reference the report.** Using "rpt" to reference the report is also possible but it is recommended to use the "this" and "Me" keywords.

 **Note:** The basic approach of using the "this/Me" and "rpt" keywords is as follows - use "this/Me" to access the properties and controls added to the sections of the report, whereas use "rpt" within the instance of the ActiveReports class only to access its public properties, public events and public methods.

- **Use error handling.** When working with script, use error handling around the .Run() call. When errors are raised, the returned error points to the section of script causing the error.

Difference in script and code-behind event handler method definition

Code-behind and the script editor require a different syntax for the event handler method definition. You should use the **Private** modifier in code-behind and the **Public** modifier in the script editor.

See the examples of the ReportStart event handler definition in Visual Basic and C# below:

Script and code-behind examples in Visual Basic

The ReportStart event handler definition in the script editor:

```
Sub ActiveReport_ReportStart
End Sub
```

The ReportStart event handler definition in code-behind:

```
Private Sub MyActiveReports_ReportStart(ByVal sender As System.Object, ByVal e As System
.EventArgs) Handles MyBase.ReportStart
End Sub
```

Script and code-behind examples in C#

The ReportStart event handler definition in the script editor:

```
public void ActiveReport_ReportStart()
{
}
```

The ReportStart event handler definition in code-behind:

```
private void MyActiveReports_ReportStart(object sender, EventArgs e)
{
}
```

Export Filters

ActiveReports provides custom components for exporting reports into six formats. Each export format has special features, however, not all formats support all of the features that you can use in your reports. Here are the unique usage possibilities of each format, along with any limitations inherent in each.

- **HTML**
- **PDF**
- **RTF**
- **Text**
- **TIFF**
- **Excel**

For information on using these export filters, see the **Export Reports** or **Custom Web Exporting (Std Edition) (on-line documentation)** topics.

HTML

HTML, or hypertext markup language, is a format that opens in a Web browser. The HTML export filter has a number of useful properties that allow you to control your output. You can set the properties either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.

Table of HTML Export Properties

| Property | Valid Values | Description |
|--------------------|--|---|
| BookmarkStyle | Html (default) or None | Set to Html to generate a page of bookmarks from the bookmarks in the report. If the report has no bookmarks, this setting is ignored. |
| CharacterSet | Big5, EucJp, HzGb2312, Ibm850, Iso2022Jp, Iso2022Kr, Iso8859_1, Iso8859_2, Iso8859_5, Iso8859_6, Koi8r, Ksc5601, ShiftJis, UnicodeUtf16, UnicodeUtf8 (default) | Select the IANA character set that you want to use in the meta tag in the header section of the HTML output. This property only takes effect if the IncludeHtmlHeader property is set to True. |
| CreateFramesetPage | True or False (default) | Set to True to generate a set of frames that display a page of bookmarks (if available) in the left frame and the report document in the right frame. The HTML output uses the specified filename with the extension .frame.html . |
| IncludeHtmlHeader | True (default) or False | Set to False if you want to embed the HTML output in another HTML document. Otherwise, the HTML output includes the usual HTML, HEAD, and BODY elements. |
| IncludePageMargins | True or False (default) | Set to True to include the report's margins in the HTML output. |
| MultiPage | True or False (default) | Set to True to create a separate HTML page for each page of the report. Otherwise, the HTML output is a single page. |
| OutputType | DynamicHtml (default) or LegacyHtml | Set to LegacyHtml to use tables for positioning and avoid the use of cascading style sheets (CSS). Otherwise, positioning of controls is handled in the |

| | | |
|---------------------|-------------------------|---|
| RemoveVerticalSpace | True or False (default) | CSS. Set to True if the OutputType property is set to LegacyHtml and you plan to print the output from a browser. This removes white space from the report to help improve pagination. Otherwise, vertical white space is kept intact. |
| Title | Any String | Enter the text to use in the header section's title. This is displayed in the title bar of the browser. |

More information on output types

By default, the report is exported as DynamicHtml (DHTML), with cascading style sheets (CSS). Using the OutputType property, you can change the output to LegacyHtml (HTML). Neither of the output types creates a report that looks exactly like the one you display in the viewer because of differences in the formats. See below for the usage of each type, and controls to avoid in each.

DynamicHtml (DHTML)

Usage:

- Create Web reports with Cascading Style Sheets (CSS)
- Open in Web browsers

Does not support:

- Diagonal line control
- CrossSectionBox control
- Control borders
- Shapes (other than filled rectangles)

LegacyHtml (HTML)

Usage:

- Create archival reports
- Open in Web browsers

Does not support:

- Line control
- Control borders
- Shapes (other than filled rectangles)
- CrossSectionBox and CrossSectionLine controls
- Overlapping controls

PDF

PDF, or portable document format, opens in the Adobe Acrobat Reader. The PDF export filter has a number of useful properties that allow you to control your output. You can set the properties either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.

Table of PDF Export Properties

| Property | Valid Values | Description |
|------------------|---------------|---|
| ConvertMetaToPng | True or False | Set to True to change any Windows metafile images to PNG format to keep the |

| | | |
|-----------------|---|---|
| | (default) | file size down. If the report has no metafiles, this setting is ignored. |
| ExportBookmarks | True (default) or False | Set to True to generate bookmarks from the bookmarks in the report. If the report has no bookmarks, this setting is ignored. To control how the exported bookmarks are displayed, use Options.DisplayMode detailed below. |
| ImageQuality | Lowest, Medium (default), or Highest | Set to Highest in combination with a high value in the ImageResolution property to yield the best printing results when converting Windows metafiles (.wmf and .emf). Set to Lowest to keep the file size down. If the report has no metafiles, this setting is ignored. |
| ImageResolution | 75 - 2400 dpi | Set to 75 dpi to save space, 150 dpi for normal screen viewing, and 300 dpi or higher for print quality. Use this property in combination with ImageQuality (highest) to yield the best results when the report contains metafiles or the Page.DrawPicture API is used. Neither property has any effect on other image types. |
| NeverEmbedFonts | A semicolon-delimited string of font names | List all of the fonts that you do not want to embed in the PDF file to keep the file size down. This can make a big difference if you use a lot of fonts in your reports. |
| Options | See below | Expand this property to see a group of subproperties. These settings control how the Adobe Acrobat Reader displays the output PDF file when it is first opened. See the table below for details. |
| Security | See below | Expand this property to see a group of subproperties. These settings control encryption and permissions on the output PDF file. See the table below for details. |
| Signature | A valid PdfSignature object. | This must be set up in code. For more information, see Digital Signatures (Pro Edition) and Create a Digital Signature for a PDF Export . |
| Version | Pdf11, Pdf12, or Pdf13 (default) | The default value is PDF specification 1.3, which is the native file format of Acrobat 4.0, or you can set it to an earlier version. Any version opens in newer Acrobat Readers. |

PDF (Portable Document Format)

Usage:

- Create printable reports whose formats do not change from machine to machine
- Open in Adobe Acrobat Reader

Does not support:

- All controls are supported
- Some limitations under **Medium trust** can be worked around using a **Custom Font Factory (Pro Edition)**
- Dash and dot border patterns appear to look longer in the PDF output than in the ActiveReports Window Forms Viewer

Options and Security

When you expand the Options or Security properties in the Properties window, the following subproperties are revealed.

Table of PDF Options Properties

| Property | Valid Values | Description |
|-------------|--------------|--|
| Application | String | Set to the string value that you want to display in the Acrobat Document |

| | | |
|--------------|---|---|
| | | Properties dialog, Description tab, Application field. |
| Author | String | Set to the string value that you want to display in the Acrobat Document Properties dialog, Description tab, Author field. |
| CenterWindow | True or False (default) | Set to True to position the Acrobat Reader window in the center of the screen when the document is first opened. |
| DisplayMode | None (default), Outlines, Thumbs, or FullScreen | Select how to display bookmarks when the document is first opened. <ul style="list-style-type: none"> • None (default) bookmarks are not displayed until opened by the user. • Outlines shows bookmarks in outline format. • Thumbs shows bookmarks as thumbnails. • FullScreen shows the document in full screen, and bookmarks are not displayed. |
| DisplayTitle | True or False (default) | Set to True to use the Title string entered in the Title property below. Otherwise, the file name is used. |
| FitWindow | True or False (default) | Set to True to expand the window to fit the size of the first displayed page. |
| HideMenubar | True or False (default) | Set to True to hide the menu in the Acrobat Reader when the document is first opened. |
| HideToolbar | True or False (default) | Set to True to hide the toolbars in the Acrobat Reader when the document is first opened. |
| HideWindowUI | True or False (default) | Set to True to hide the scrollbars and navigation controls in the Acrobat Reader when the document is first opened, displaying only the document. |
| Keywords | String | Enter keywords to display in the Acrobat Document Properties dialog, Description tab, Keywords field. |
| Subject | String | Enter a subject to display in the Acrobat Document Properties dialog, Description tab, Subject field. |
| Title | String | Enter a title to display in the Acrobat Document Properties dialog, Description tab, Title field. Set DisplayTitle to True to display this text in the title bar of the Acrobat Reader when the document is opened. |

Table of PDF Security Properties

| Property | Valid Values | Description |
|---------------|---|---|
| Encrypt | True or False (default) | Sets or returns a value indicating whether the document is encrypted. |
| OwnerPassword | String | Enter the string to use as a password that unlocks the document regardless of specified permissions. |
| Permissions | None, AllowPrint, AllowModifyContents, AllowCopy, AllowModifyAnnotations, AllowFillIn, AllowAccessibleReaders, or AllowAssembly | Combine multiple values by dropping down the selector and selecting the check boxes of any permissions you want to grant. By default, all of the permissions are granted. |
| Use128Bit | True (default) or False | Set to False to use 40 bit encryption with limited permissions. (Disables AllowFillIn, AllowAccessibleReaders, and AllowAssembly permissions.) |
| UserPassword | String | Enter the string to use as a password that unlocks the |

document using the specified permissions. Leave this value blank to allow anyone to open the document using the specified permissions.

Digital Signatures (Pro Edition)


With the Professional Edition license, you can digitally sign a report when exporting it to PDF format. The digital signature identifies by whom, when and for what reason the document was created, and sets the certification level that dictates how the document may be accessed and modified by other users. A digital signature serves as means of protecting the document from any unauthorized access, use or modification.

In order to use digital signatures, you must first have a valid PKCS#12 certificate (*.pfx) file. You can use a third-party digital ID, or create a self-signed certificate.

For information on creating a self-signed certificate, see the [Adobe Acrobat Help topic "Create a self-signed digital ID."](#)

You can also create a PFX file from the Visual Studio command line. For more information and links to SDK downloads, see <http://www.source-code.biz/snippets/vbasic/3.htm>.

A digital signature may be *invisible* or *visible*, the latter containing *text* or *graphics* elements. It is displayed under the **Signatures** tab on the left side of the PDF document window.

 **Note:** You cannot digitally sign a report if it is exported into a password-protected pdf document.

To learn more about PDF digital signatures, please refer to www.adobe.com/support/documentation.

Table of Property Descriptions

| Property Name | Description |
|---------------------------------|---|
| export.Signature.Certificate | Sets the certificate for the digital signature. |
| export.Signature.VisibilityType | Sets the signature type: visible or invisible. E.g., the value for a visible signature, containing text and graphics will be = <code>VisibilityType.ImageText</code> . If the property is not set or set to "Invisible" expressly, the signature will be invisible regardless of any other setting. |
| export.Signature.SignDate | Specifies the time of signing the document (e.g. = <code>DateTime.Now</code>). |
| export.Signature.Contact | Specifies the signature contact information. |
| export.Signature.Reason | Specifies the signature reason information. |
| export.Signature.Location | Specifies the signature location information. |
| export.Signature.TimeStamp | Specifies the settings for the signature time stamp: the Time Stamp Server address, its login and password information (e.g. = <code>New TimeStamp("http://free-tsu.e-timing.ne.jp/TSS/HttpTspServer", "null", "null")</code>). |
| export.Signature.Stamp.Bounds | Sets the bounds for the signature display (e.g. = <code>new RectangleF(1, 1, 4, 2)</code>).The unit of measure is inches. The upper-left corner is a start point of the signature rectangle. |
| export.Signature.Stamp.Image | Specifies the image settings in case the signature contains |

| | |
|---|--|
| <code>graphics (e.g. = Image.FromFile("image.png")).</code> | |
| <code>export.Signature.Stamp.TextRectangle</code> or <code>export.Signature.Stamp.ImageRectangle</code> | Specify the area where an image or text will be placed inside the signature rectangle (e.g. = <code>New RectangleF(0, 0.135, 3, 1)</code>). The property uses the upper-left corner as a start point and is specified in coordinates, relative to the signature rectangle. If this property is not specified, the entire signature rectangle will be used for placing an image or text. |
| <code>export.Signature.Stamp.TextAlignment</code> | Specifies whether the text in the signature is left-aligned, right-aligned, or centered. (e.g. = <code>Alignment.Left</code>).The alignment is performed inside the text rectangle that is included into the signature rectangle. |
| <code>export.Signature.Stamp.ImageAlignment</code> | Specifies the alignment of an image inside the image rectangle that is included into the signature rectangle.(e.g. = <code>Alignment.Right</code>). |
| <code>export.Signature.Stamp.Font</code> | Specifies the signature stamp font (e.g. = <code>new Font("Arial", 10, FontStyle.Italic)</code>). |
| <code>export.Signature.Stamp.TextColor</code> | Specifies the stamp text color (e.g. = <code>Color.AliceBlue</code>). |
| <code>export.Signature.CertificationLevel</code> | Sets the level of other users' access to the document (e.g. = <code>CertificationLevel.FormFilling</code>). |

 **Note:** Be careful when changing fonts of exports using localization. If a selected font does not support the language, then the localized labels are not shown in the signature.

Custom Font Factory (Pro Edition)

When you use the PDF export filter in a Medium trust environment, ActiveReports does not have access to the System Fonts folder. So if your reports use special glyphs or non-ASCII characters that are only found in certain fonts, the PDF output may be incorrect on some machines. To deploy the necessary fonts with your Medium trust solution, you can create a custom font factory in your web.config file.

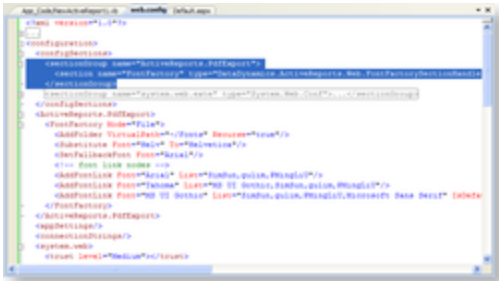
To add a font factory section group

Paste between the `<configSections>` and `</configSections>` tags.

XML code. Paste INSIDE the configSections tags.

```
<sectionGroup name="ActiveReports.PdfExport">
  <section name="FontFactory"

type="DataDynamics.ActiveReports.Web.FontFactorySectionHandler,ActiveReports.Web,
  Version=6.0.2227.0, Culture=neutral,PublicKeyToken=cc496777c49a3ff"
  requirePermission="false"/>
</sectionGroup>
```




To create a font factory

Paste code like the following after the </configSections> tag, but before the <appSettings/> tag.

XML code. Paste between configSections and appSettings.

```
<ActiveReports.PdfExport>
  <FontFactory Mode="File">
    <AddFolder VirtualPath="~/Fonts" Recurse="true"/>
    <Substitute Font="Helv" To="Helvetica"/>
    <SetFallbackFont Font="Arial"/>
    <!-- font link nodes -->
    <AddFontLink Font="Arial" List="SimSun,gulim,PMingLiU"/>
    <AddFontLink Font="Tahoma" List="MS UI Gothic,SimSun,gulim,PMingLiU"/>
    <AddFontLink Font="MS UI Gothic" List="SimSun,gulim,PMingLiU,Microsoft Sans
Serif" IsDefault="true"/>
  </FontFactory>
</ActiveReports.PdfExport>
```



 **Note:** For the Azure **worker role** project, use an absolute path instead of a virtual path in the code above:

```
<AddFolder Path="~/Fonts" Recurse="true"/>.
```

To set up a Medium trust environment

Paste the following code between the <system.web> and </system.web> tags.

XML code. Paste BETWEEN the system.web tags.

```
<trust level="Medium"></trust>
```

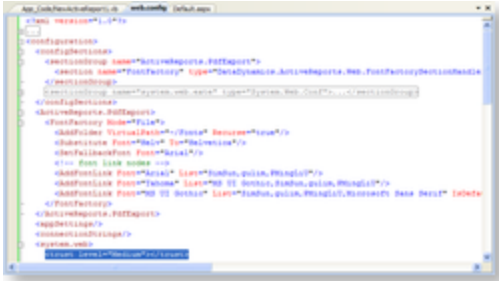




Table of configuration settings

| Setting | Description | Attributes |
|---|---|--|
| FontFactory | This is the main font factory node to which you can add fonts. | Mode: Set to "File" to use a file based factory, or remove the attribute for a Windows GDI factory. |
| AddFolder | Adds all TrueType fonts from the specified folder. | Path: Specify the absolute path to the folder. VirtualPath: Specify the relative path to the folder. Recurse: Set to "true" to have the factory read folders recursively. |
| Substitute | Maps alternate spellings of fonts to their official names. | Font: The abbreviated font name (e.g. "Helv"). To: The official font name (e.g. "Helvetica"). |
| SetFallbackFont Professional Edition only | In the Professional Edition, sets the font to use in cases where: <ol style="list-style-type: none"> the specified font is not installed, the Substitute font is not specified or not installed, the font links are not set or the needed glyphs are not found in the AddFontLink setting. | Font: The font name. |
| AddFontLink Professional Edition only | In the Professional Edition, there is extra support for CJK glyphs. You can add font links that allow the PdfExport to look up any glyphs missing from the specified Font in the List of other fonts to check. | Font: The font used in the reports. List: The comma-separated list of fonts in which to look for missing glyphs. IsDefault: Set this to "true" to use the specified List for any fonts that do not have their own font links set. |

 **Note:** EUDC fonts are not yet supported in "File" mode.

 **Note:** For additional information on the **SetFallbackFont** and **AddFontLink** settings, see **Font Linking**.


 **Note:** For the Azure project, set the properties for all fonts in the project Fonts folder as follows:

1. Set **Copy to Output Directory** to **Copy always**.
2. Set **BuildAction** to **Content**.


Font Linking (Pro Edition)

Fonts on a deployment machine may not have the glyphs that were used in a development environment. In case of the glyphs mismatch, the PDF output on development and deployment machines may be different. In order to resolve this limitation, the PDF export filter looks for the missing glyphs in the installed fonts in the order as follows:

Firstly, the export filter checks the system font link settings for each font that was used in the report.

 **Note:** HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FontLink\SystemLink registry key stores the information on font links.

If font links are not set or the needed glyphs are not found, the export filter looks for the glyphs in the fonts declared in the **FontFallback property (on-line documentation)**. Further on, the export filter can use glyphs from the font links collection to replace fonts that do not have own declared linked fonts.

 **Note:** If necessary glyphs were not found by font links or in the fonts declared in the FontFallback property, the PDF export filter takes the glyphs from the predefined Microsoft Sans Serif font.

RTF

RTF, or RichText format, opens in Microsoft Word, and is native to WordPad. This export does not render reports exactly as they appear in the Viewer due to inherent differences in the formats.

The RTF export filter has one property, **EnableShapes**, that allows you to control your output. You can set the property either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.


Usage:

- Create word-processing files
- Open in Word or WordPad

Does not support:

- Section Backcolors
- Overlapping controls
- Angled text

Only supported when EnableShapes property is True:

 If you set the **EnableShapes** property to **True**, the resulting RTF file displays correctly only in **Web Layout View** in Microsoft Word.

- Full justification
- Line control
- Control Backcolors
- Shapes
- Control borders

Text

Plain Text is a format that opens in Notepad or Microsoft Excel depending on the file extension you use in the filePath parameter of the Export method. Use the extension **.txt** for files to open in Notepad, or use **.csv** for comma separated value files to open in Excel. The Text export filter has a number of useful properties that allow you to control your output. You can set the properties either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.

Table of Text Export Properties

| Property | Valid Values | Description |
|--------------------|---|---|
| Encoding | System.Text.ASCIIEncoding (default), System.Text.UnicodeEncoding, System.Text.UTF7Encoding, or System.Text.UTF8Encoding | This property can only be set in code. Enter an enumerated system encoding value to use for character encoding. |
| PageDelimiter | String | Enter a character or sequence of characters to mark the end of each page. |
| SuppressEmptyLines | True (default) or False | Set to False if you want to keep empty lines in the exported text file. Otherwise, white space is removed. |
| TextDelimiter | String | Enter a character or sequence of characters to mark the end of each text field. This is mainly for use with CSV files that you open in Excel. |

Text

Usage:

- Create plain text files
- Create comma (or other character) delimited text files
- Feed raw data to spreadsheets or databases
- Open in Notepad or Excel (comma delimited)

Does not support anything but plain fields and labels:

- Supports plain text only with no formatting other than simple delimiters
- Supports encoding for foreign language support

TIFF

TIFF, or tag image file format, opens in the Windows Picture and Fax Viewer or any TIFF viewer. This export looks very much like the report as it displays in the viewer, but it is a multi-page image, so the text cannot be edited. The TIFF export filter has a couple of useful properties that allow you to control your output. You can set the properties either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.

Table of TIFF Export Properties

| Property | Valid Values | Description |
|-------------------|--|--|
| CompressionScheme | None, Rle, Ccitt3 (default), Ccitt4 or Lzw | Select an enumerated value to use for color output control: <ul style="list-style-type: none"> • None delivers color output with no compression. • Rle (run-length encoding) is for 1, 4, and 8 bit color depths. • Ccitt3 and Ccitt4 are for 1 color depth, and are used in old standard faxes. • Lzw (based on Unisys patent) is for 1, 4, and 8 bit color depths with lossless compression. |
| Dither | True or False (default) | Set to True to dither the image when you save it to a black and white format (Ccitt3, Ccitt4 or |

| | | |
|------|---|--|
| DpiX | Integer (VB) or int (C#) greater than 0 | <p>Rle). This property has no effect if the CompressionScheme is set to Lzw or None.</p> <p>Set the horizontal resolution of a report when exporting to TIFF format. The default value is 200.</p> <p>Setting the DpiX or DpiY property to large values can cause the rendered image to be too large and not enough memory in system can be allocated to the bitmap.</p> |
| DpiY | Integer (VB) or int (C#) greater than 0 | <p>Set the vertical resolution of a report when exporting to TIFF format. The default value is 196.</p> <p>Setting the DpiX or DpiY property to large values can cause the rendered image to be too large and not enough memory in system can be allocated to the bitmap.</p> |

Usage:

- Create optical archive reports
- Send reports via fax machines
- Open in image viewers

Image of each page, so it's 100% WYSIWYG.

Excel

XLS is a format that opens in Microsoft Excel as a spreadsheet. This export does not render reports exactly as they appear in the Viewer due to inherent differences in the formats. The XLS export filter has a number of useful properties that allow you to control your output. You can set the properties either in code using the export object, or by selecting the object in the component tray below the form and using the Properties window.

Table of XLS Export Properties

| Property | Valid Values | Description |
|------------------|------------------------------|---|
| AutoRowHeight | True or False (default) | Set to True to have Excel set the height of rows based on the contents. Otherwise XlsExport calculates the height of rows. In some cases this may make the output look better inside Excel. However, a value of True may adversely affect pagination when printing, as it may stretch the height of the page. |
| DisplayGridLines | True (default) or False | Set to False to hide grid lines in Excel. |
| FileFormat | Xls97Plus (default) or Xls95 | Set to Xls95 to use Microsoft Excel 95 format. Otherwise, a format optimized for Excel 97 and newer is used. |
| MinColumnWidth | Single (VB) or float (C#) | Set the number of inches that is the smallest width for a column in the exported spreadsheet. Tip: Larger values reduce the number of empty columns in a sheet. Set this value to 1 inch or more to get rid of small empty columns. |

| | | |
|---------------------|---------------------------|--|
| MinRowHeight | Single (VB) or float (C#) | Set the number of inches that is the smallest height for a row in the exported spreadsheet. Tip: Larger values force the export to place more controls on a single line by reducing the number of rows added to match blank space. Set this value to .25 inches or more to get rid of small empty rows. |
| MultiSheet | True or False (default) | Set to True to export each page of your report to a separate sheet within the Excel file. This can increase performance and output quality at the cost of memory consumption for reports with complex pages and a lot of deviation between page layouts. In general, use False for reports with more than 30 pages. |
| RemoveVerticalSpace | True or False (default) | Set to True to remove vertical empty spaces from the spreadsheet. This may improve pagination for printing. |
| UseCellMerging | True or False (default) | Set to True to merge cells where applicable. |

TextBox Numeric Values Export

The **OutputFormat** property of the **TextBox** control supports the following numeric types:

- Int16
- Int32
- Int64
- UInt16
- UInt32
- UInt64
- Single
- Double
- Decimal

A numeric value of the **TextBox** control is exported to Excel as a number if the **Value** property is set to a supported numeric type and the **OutputFormat** property is set to a valid value.

TextBox Date Values Export

A date value of the **TextBox** control is exported to Excel as a date if the **Value** property is set to a supported date type and the **OutputFormat** property is set to a valid value.

The numeric and date value of the **TextBox** control is exported to Excel as text in the following cases:

- the **OutputFormat** property is null or empty.
- the **Value** property is not set to any of the supported numeric types.
- the **Value** property is set to a decimal, long or ulong value and the number of digits is larger than 15.
- the **Value** property is set to a supported numeric value, but the **OutputFormat** property is set to an invalid format in Excel.
- the **Value** property is set to the date before 1900/3/1.

A numeric value of the **Label** and **RichTextBox** controls is always exported to Excel as text.

Usage:

- Create spreadsheets
- Open in Microsoft Excel

Does not support:

- Line control
- Shapes (other than filled rectangles)
- CrossSectionBox and CrossSectionLine controls
- Overlapping controls
- Borders on controls with angled text
- Angled text
- CheckBox control (only its text element is exported)

Charts

Follow the links below for information about concepts essential to the use of the Chart control.

Chart Elements

See an overview of the different pieces that make up an ActiveReports Chart.

Chart Series

Explains what a series is and how it comprises the data that is seen on a chart.

Chart- and Series-Specific Properties

Learn which series properties apply to each of the Chart Types.

Chart Wizard

Shows how to access the Chart Wizard.

Chart Types

Shows examples of Common Charts, 3D Charts, XY Charts, and Financial Charts.

Chart Appearance

Covers Chart Effects, Chart Control Items, and Chart Axes and Walls.

Chart Data

Discusses ways of connecting a chart to data.

Chart Elements

You can use the ActiveReports Chart control to display data visually and help readers to easily analyze and interpret numerical and relational data. The elements that make up an ActiveReports chart bring meaning to the visual information. You have the following major elements at your disposal:

- Axes
- Series
- Points
- Titles
- Legends

The following image illustrates the elements that make up the ActiveReports Chart control.

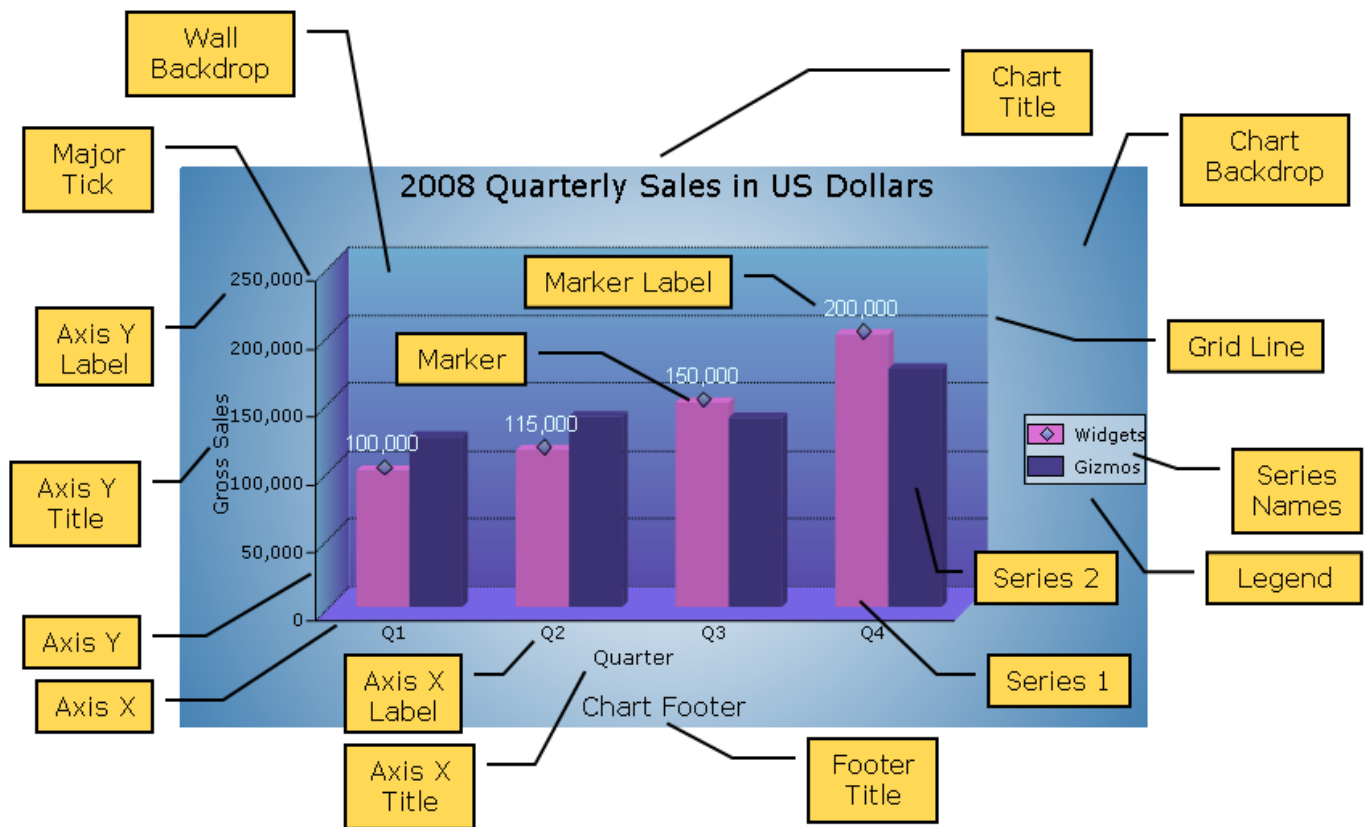


Chart Elements

[Axis Label](#)

A label along an axis that lets you label the units being shown.

[Axis Title](#)

The axis title allows you to provide a title for the information being shown on the axis.

[Chart Backdrop](#)

The chart backdrop is the background for the whole chart that is created. You can create your own backdrop using the different styles and colors available or you can use an image as a backdrop for your chart.

[Chart Title](#)

The chart title serves as the title for the chart control.

[Footer Title](#)

The footer title allows you to add a secondary title for the chart control along the bottom.

[Grid Line](#)

Grid lines can occur on horizontal and vertical axes and normally correlate to the major or minor tick marks for the axes.

[Legend](#)

The legend serves as a key to the specific colors or patterns being used to show series values in the chart.

[Marker](#)

The marker is used to annotate a specific plotted point in a data series.

[Marker Label](#)

The marker label allows you to display the value of a specific plotted point in a data series.

[Major Tick](#)

Major tick marks can occur on horizontal and vertical axes and normally correlate to the major gridlines for the axes.

[Minor Tick](#)

Minor tick marks can occur on horizontal and vertical axes and normally correlate to the minor gridlines for the axes.

[Series](#)

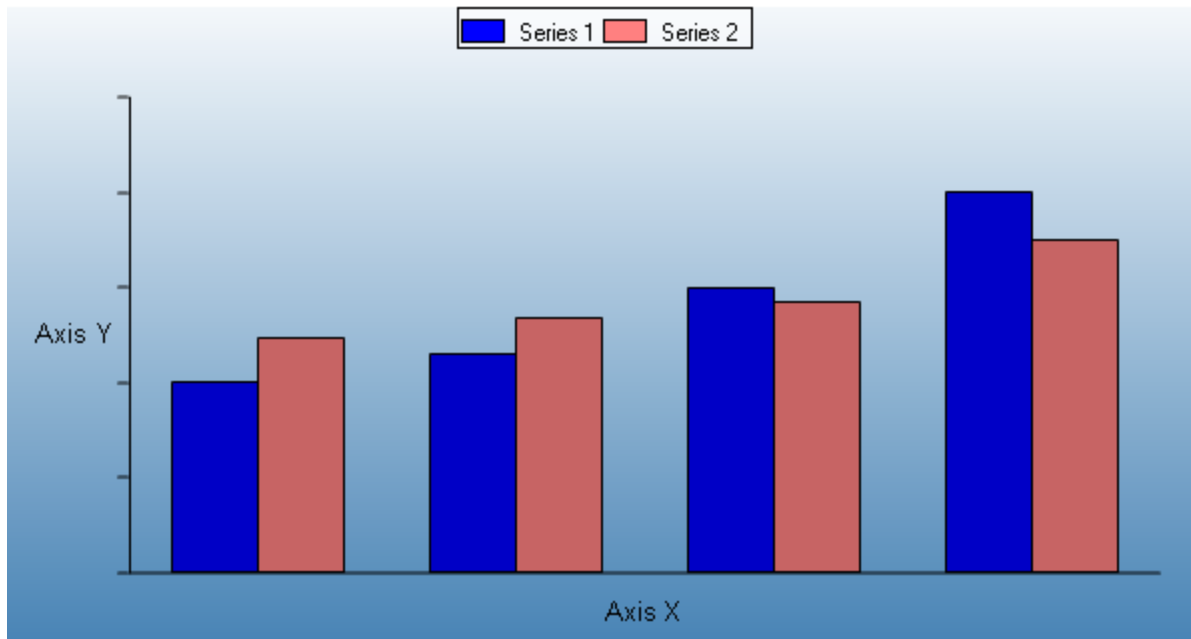
The series is a related group of data values that are plotted on the chart. Each plotted point is a data point that reflects the specific values charted. Most charts, such as the above bar chart, can contain more than one series, while others, such as a pie chart, can contain only one.

[Wall Backdrop](#)

The wall is the back section of the chart on which data is plotted.

Chart Series

A chart series is the key to showing data in a chart. All data is plotted in a chart as a series and all charts contain at least one series. The bars in the image below depict two series in a simple bar chart.



Each series is made up of a set of data points consisting of an X value that determines where on the X axis the data is plotted, and one or more Y values. Most charts use one Y value but a few charts such as the Bubble, BubbleXY, and the financial charts take multiple Y values.

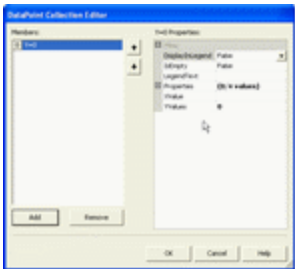
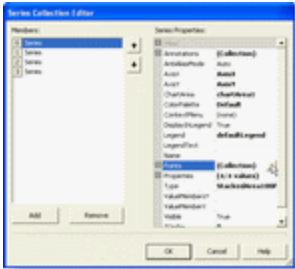
When you bind data to a series, the X value is bound using the ValueMembersX property on the Series object, and the Y value is bound using the ValueMembersY property.

The Series object also contains properties for each individual series, including chart type, custom chart properties, annotations, containing chart area, and more. Each chart type in the ActiveReports Chart control contains series-specific properties that apply to it. You can set the chart type and these series-specific properties in the Series Collection Editor dialog, which opens when you click the ellipsis button next to the **Series (Collection)** property in the Visual

Studio Properties window.



You can manipulate each data point in the DataPoint Collection dialog box. You can access the dialog from the Series Collection Editor by clicking the ellipsis button next to the **Points (Collection)** property.



When you set a property on the Series object, it is applied to all data point objects in the series unless a different value for the property is set on a specific data point. In that case, the data point property setting overrides the series property setting for that particular data point. Note that for charts bound to a data source, you do not have access to the DataPoint collection in the dialog.

If you specify the value for any of the custom properties, this value is not cleared when you change the ChartType. Although this will show properties that do not apply to certain ChartTypes, it has the advantage of keeping your settings in case you accidentally change the ChartType.

Setting chart and series-specific properties at run time

To set custom properties for a chart on the series programmatically, reference the series by name or index and use the string Properties attribute name you wish to set.

The following code samples set the shape for bubbles on a bubble chart to diamond.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Shape") = Chart.MarkerStyle.Diamond
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Shape"] =  
DataDynamics.ActiveReports.Chart.MarkerStyle.Diamond;
```

To set custom properties for a chart on the data points object programmatically, reference the series by name or index, reference the data point by index, and use the string Properties attribute name you wish to set.

The following code samples set the explode factor on a doughnut chart for the second point in the series.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Points(1).Properties("ExplodeFactor") = 0.5F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Points[1].Properties["ExplodeFactor"] = .5f;
```

Chart and Series-Specific Properties

Each chart type in the ActiveReports Chart control contains specific properties that apply to it. You can set the chart type and the correlating series-specific properties in the Series Collection Editor dialog via the Series (Collection) property in the Properties Window of Visual Studio and in the DataPoint Collection dialog box via the Points (Collection) property in the Series dialog.

Chart Standard Properties

- **Backdrop:** Gets or sets the backdrop information for the series. Does not apply to Bezier, Line, LineXY, Line3D, PlotXY, or Scatter charts.
- **BorderLine:** Gets or sets the line information used to draw the border of the series. Does not apply to Bezier, Line, LineXY, PlotXY, or Scatter charts.
- **Line:** Gets or sets the line information for the series. Only applies to Bezier, Line, and LineXY charts.
- **Marker:** Gets or sets the ToolTip settings for the series.
- **ToolTip:** Gets or sets the ToolTip information for the series.

Charts and Custom Properties

Area (none)

Area3D

Custom Properties

- **LineBackdrop** Gets or sets the backdrop information for the 3D line.
- **Thickness** Gets or sets the thickness of the 3D line.
- **Width** Gets or sets the width of the 3D line.

Bar2D

Custom Property

- **Gap** Gets or sets the space between the bars of each X axis value.

Bar3D

Custom Properties

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.

- **PointBarDepth** Gets or sets the thickness of the 3D bar.
- **RotationAngle** Gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Bezier

Custom Property

- **Tension** Gets or sets the tension of the curved lines. Values must be less than or equal to 1. Default is null.

Bezier3D

Custom Property

- **Tension** Gets or sets the tension of the curved lines. Values must be less than or equal to 1. Default is null.
- **Width** Gets or sets the width of the 3D line.

Bubble

Custom Properties

- **MaxSizeFactor** Gets or sets the maximum size of the bubble radius. Values must be less than or equal to 1. Default is .25.
- **MaxValue** Gets or sets the bubble size that is used as the maximum.
- **MinValue** Gets or sets the bubble size that is used as the minimum.
- **Shape** Gets or sets the shape of the bubbles. Uses or returns a valid MarkerStyle enumeration value.

Candle

Custom Properties

- **BodyDownswingBackdrop** Gets or sets the backdrop information used to fill the downswing rectangle.
- **BodyUpswingBackdrop** Gets or sets the backdrop information used to fill the upswing rectangle.
- **BodyWidth** Gets or sets the width of the rectangle used to show upswing or downswing.
- **Wickline** Gets or sets the line information for the wick line.

ClusteredBar

Custom Properties

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.
- **RotationAngle** Gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Doughnut

Custom Properties

- **Clockwise** Gets or sets a value indicating whether to display the data in clockwise order. By default, the data is displayed counterclockwise beginning at 12 o'clock.
- **ExplodeFactor** Gets or sets the amount of separation between data point values. The value must be less than or equal to 1. To explode one section of the doughnut chart, set ExplodeFactor on the data point instead of on the series.
- **HoleSize** Gets or sets the inner radius of the chart. If set to 0, the chart looks like a pie chart. The value must be less than or equal to 1.
- **OutsideLabels** Gets or sets a value indicating whether the data point labels appear outside the chart.
- **Radius** Gets or sets the size of the doughnut within the chart area.

- **StartAngle** Gets or sets the horizontal start angle for the series.

Doughnut3D

Custom Properties

- **Clockwise** Gets or sets a value indicating whether to display the data in clockwise order. By default, the data is displayed counterclockwise beginning at 12 o'clock.
- **ExplodeFactor** Gets or sets the amount of separation between data point values. The value must be less than or equal to 1. To explode one section of the doughnut chart, set ExplodeFactor on the data point instead of on the series.
- **HoleSize** Gets or sets the inner radius of the chart. If set to 0, the chart looks like a pie chart. The value must be less than or equal to 1.
- **OutsideLabels** Gets or sets a value indicating whether the data point labels appear outside the chart.
- **Radius** Gets or sets the size of the doughnut within the chart area.
- **StartAngle** Gets or sets the horizontal start angle for the series data points.

Funnel

Custom Properties

- **CalloutLine** Gets or sets the style for a line connecting the marker label to its corresponding funnel section. The default value is a black one-point line.
- **FunnelStyle** Gets or sets the Y value for the series points to the width or height of the funnel. The default value is YIsHeight.
- **MinPointHeight** Gets or sets the minimum height allowed for a data point in the funnel chart. The height is measured in relative coordinates.
- **NeckHeight** Gets or sets the neck height for the funnel chart. This property can only be used with the FunnelStyle property set to YIsHeight. The default value is 5.
- **NeckWidth** Gets or sets the neck width for the funnel chart. This property can only be used with the FunnelStyle property set to YIsHeight. The default value is 5.
- **OutsideLabels** Gets or sets a value indicating whether the labels are placed outside of the funnel chart. The default value is True.
- **OutsideLabelsPlacement** Gets or sets a value indicating whether the data point labels appear on the left or right side of the funnel. This property can only be used with the OutsideLabels property set to True.
- **PointGapPct** Gets or sets the amount of space between the data points of the funnel chart. The PointGapPct is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.

Funnel3D

Custom Properties

- **BaseStyle** Gets or sets a circular or square base drawing style for the 3D funnel chart. This property only takes effect if the Projection property is set to Orthogonal.
- **CalloutLine** Gets or sets the style for a line connecting the marker label to its corresponding funnel section. The default value is a black one-point line.
- **FunnelStyle** Gets or sets the Y value for the series points to the width or height of the funnel. The default value is YIsHeight.
- **MinPointHeight** Gets or sets the minimum height allowed for a data point in the funnel chart. The height is measured in relative coordinates.
- **NeckHeight** Gets or sets the neck height for the funnel chart. This property can only be used with the FunnelStyle property set to YIsHeight. The default value is 5.
- **NeckWidth** Gets or sets the neck width for the funnel chart. This property can only be used with the FunnelStyle property set to YIsHeight. The default value is 5.
- **OutsideLabels** Gets or sets a value indicating whether the labels are placed outside of the funnel chart. The default value is True.
- **OutsideLabelsPlacement** Gets or sets a value indicating whether the data point labels appear on the left or right side of the funnel. This property can only be used with the OutsideLabels property set to True.

- **PointGapPct** Gets or sets the amount of space between the data points of the funnel chart. The PointGapPct is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.
- **RotationAngle** Gets or sets the left-to-right rotation angle of the funnel. The valid values range from -180 to 180 degrees. This property is only effective with the Projection property set to Orthogonal and the BaseStyle property set to SquareBase.

Gantt

Custom Property

- **Gap** Gets or sets the space between the bars of each X axis value.

Gantt3D

Custom Property

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.
- **PointBarDepth** Gets or sets the thickness of the 3D bar.
- **RotationAngle** Gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

HiLo

Custom Property

- **HiLoLine** Gets or sets the line information for the hi-lo line.

HiLoOpenClose

Custom Properties

- **CloseLine** Gets or sets the line information for the close line.
- **HiLoLine** Gets or sets the line information for the hi-lo line.
- **OpenLine** Gets or sets the line information for the open line.
- **TickLen** Gets or sets the length of the tick for the open and close lines.

HorizontalBar

Custom Property

- **Gap** Gets or sets the space between the bars of each X axis value.

HorizontalBar3D

Custom Properties

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.
- **PointBarDepth** Gets or sets the thickness of the 3D bar.
- **RotationAngle** Gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Kagi

Custom Properties

- **DownswingLine** Gets or sets the style and color settings to use for a Kagi line which charts a price decrease.

- **ReversalAmount** Gets or sets the amount that a price must shift in order for the Kagi line to change direction.
- **UpswingLine** Gets or sets the style and color settings to use for a Kagi line which charts a price increase.

Kagi3D

Custom Properties

- **BodyDownswingBackdrop** Gets or sets the style and color settings for the three-dimensional side view of downswing Kagi lines. This property is only effective when the Width property is set to a value higher than 25.
- **BodyUpswingBackdrop** Gets or sets the style and color settings for the three-dimensional side view of upswing Kagi lines. This property is only effective when the Width property is set to a value higher than 25.
- **DownswingLine** Gets or sets the style and color settings to use for a Kagi line which charts a price decrease.
- **ReversalAmount** Gets or sets the amount that a price must shift in order for the Kagi line to change direction.
- **UpswingLine** Gets or sets the style and color settings to use for a Kagi line which charts a price increase.
- **Width** Gets or sets the width of the three-dimensional side view of the Kagi lines. This property must be set higher than its default value of 1 in order to display body downswing and upswing backdrops.

Line

Custom Property

- **LineJoin** Gets or sets the type of join to draw when two lines connect. Valid values include Miter, Bevel, Round, and MiterClipped.

Line3D

Custom Properties

- **LineBackdrop** Gets or sets the backdrop information for the 3D line.
- **Thickness** Gets or sets the thickness of the 3D line.
- **Width** Gets or sets the width of the 3D line.

Point and Figure

Custom Properties

- **BoxSize** Gets or sets the amount a price must change in order to create another X or O.
- **DownswingLine** Gets or sets the style and color settings for the downswing Os.
- **ReversalAmount** Gets or sets the amount that a price must shift in order for a new column to be added.
- **UpswingLine** Gets or sets the style and color settings for the upswing Xs.

Pyramid

Custom Properties

- **CalloutLine** Gets or sets the style for a line connecting the marker label to its corresponding pyramid section. The default value is a black one-point line.
- **MinPointHeight** Gets or sets the minimum height allowed for a data point in the pyramid chart. The height is measured in relative coordinates.
- **OutsideLabels** Gets or sets a value indicating whether the labels are placed outside of the pyramid chart. The default value is True.
- **OutsideLabelsPlacement** Gets or sets a value indicating whether the data point labels appear on the left or right side of the pyramid. This property can only be used with the OutsideLabels property set to True.
- **PointGapPct** Gets or sets the amount of space between the data points of the pyramid chart. The PointGapPct is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.

Pyramid3D

Custom Properties

- **BaseStyle** Gets or sets a circular or square base drawing style for the 3D pyramid chart. This property only takes effect with the Projection property set to Orthogonal.

- **CalloutLine** Gets or sets the style for a line connecting the marker label to its corresponding pyramid section. The default value is a black one-point line.
- **MinPointHeight** Gets or sets the minimum height allowed for a data point in the pyramid chart. The height is measured in relative coordinates.
- **OutsideLabels** Gets or sets a value indicating whether the labels are placed outside of the pyramid chart. The default value is True.
- **OutsideLabelsPlacement** Gets or sets a value indicating whether the data point labels appear on the left or right side of the pyramid. This property can only be used with the OutsideLabels property set to True.
- **PointGapPct** Gets or sets the amount of space between the data points of the pyramid chart. The PointGapPct is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.
- **RotationAngle** Gets or sets the left-to-right rotation angle of the pyramid. The valid values range from -180 to 180 degrees. This property is only effective with the Projection property set to Orthogonal and the BaseStyle property set to SquareBase.

Renko

Custom Properties

- **BodyDownswingBackdrop** Gets or sets the style and color settings for the downswing bricks.
- **BodyUpswingBackdrop** Gets or sets the style and color settings for the upswing bricks.
- **BoxSize** Gets or sets the amount a price must change in order to create another brick.

Renko3D

Custom Properties

- **BodyDownswingBackdrop** Gets or sets the style and color settings for the downswing bricks.
- **BodyUpswingBackdrop** Gets or sets the style and color settings for the upswing bricks.
- **BoxSize** Gets or sets the amount a price must change in order to create another brick.

Scatter (none)

StackedArea (none)

StackedArea3D

Custom Property

- **Width** Gets or sets the width of the 3D area.

StackedBar

Custom Property

- **Gap** Gets or sets the space between the bars of each X axis value.

StackedBar3D

Custom Properties

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

StackedArea100Pct (none)

StackedArea3D100Pct

Custom Property

- **Width** Gets or sets the width of the 3D area.

StackedBar100Pct

Custom Property

- **Gap** Gets or sets the space between the bars of each X axis value.

StackedBar3D100Pct**Custom Properties**

- **BarTopPercent** Gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
- **BarType** Gets or sets the type of bars that are displayed. Values are Bar, Cylinder, Cone, Pyramid, and Custom.
- **Gap** Gets or sets the space between the bars of each X axis value.
- **VertexNumber** Gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Three Line Break**Custom Properties**

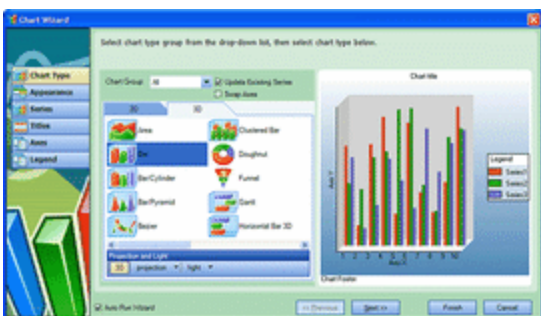
- **BodyDownswingBackdrop** Gets or sets the style and color settings for the downswing boxes.
- **BodyUpswingBackdrop** Gets or sets the style and color settings for the upswing boxes.
- **NewLineBreak** Gets or sets the number of previous boxes/lines that must be compared before a new box/line is drawn. The default value is 3.

Three Line Break3D**Custom Properties**

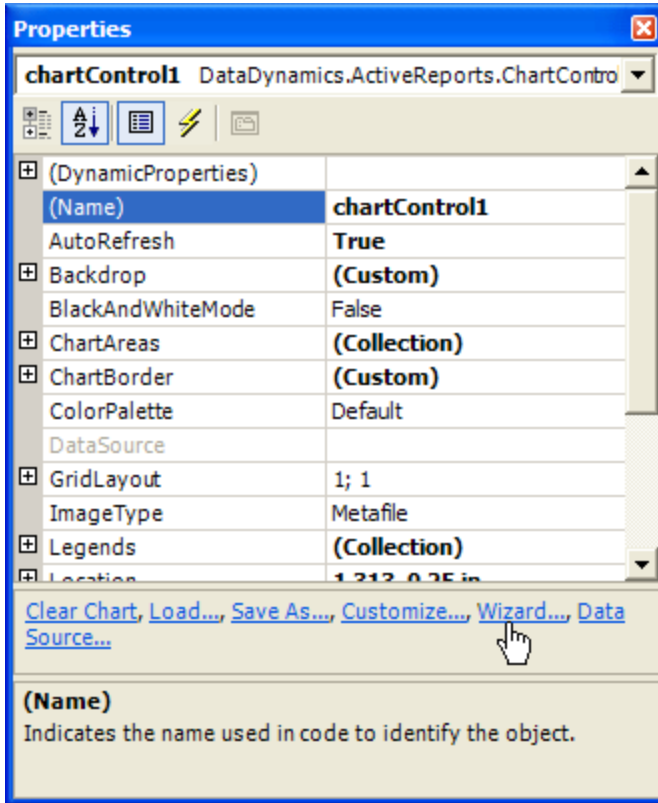
- **BodyDownswingBackdrop** Gets or sets the style and color settings for the downswing boxes.
- **BodyUpswingBackdrop** Gets or sets the style and color settings for the upswing boxes.
- **NewLineBreak** Gets or sets the number of previous boxes/lines that must be compared before a new box/line is drawn. The default value is 3.

Chart Wizard

The chart control features an easy-to-use wizard. The chart wizard automatically runs when you first add a chart control to a report. If you prefer not to have the wizard run automatically, clear the Auto Run Wizard checkbox at the bottom of the wizard.



You can also access the wizard through the Wizard verb that appears below the Properties window when the chart is selected on the report.



If the verb does not appear in the Properties window, see **Access the Chart Wizard and Data Source**.

Chart Types

These topics introduce you to the different Chart Types you can create with the Chart control.



Common Charts

Area, Bar2D, Bezier, Doughnut/Pie, Line, Scatter, StackedArea, StackedBar, StackedArea100Pct, and StackedBar100Pct

3D Charts

Area3D, Bar3D, ClusteredBar, Line3D, Doughnut3D/Pie, StackedBar3D, and StackedBar3D100Pct

XY Charts

Bubble, BubbleXY, LineXY, and PlotXY

Financial Charts

Candle, HiLo, and HiLoOpenClose

Common Charts

The ActiveReports Chart control can draw a number of 2D chart types:

- Area, Bar 2D, Bezier
- Doughnut and Pie, Gantt
- Horizontal Bar, Line
- Scatter, Stacked Area, Stacked Area 100 Percent
- Stacked Bar, Stacked Bar 100 Percent

See below for details on each of the common chart types.

Area Chart

Use an area chart to compare trends over a period of time or across categories.

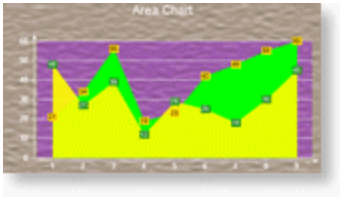


Chart Information

Chart Information

| | |
|--|----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | None |

Bar 2D Chart

Use a bar chart to compare values of items across categories.

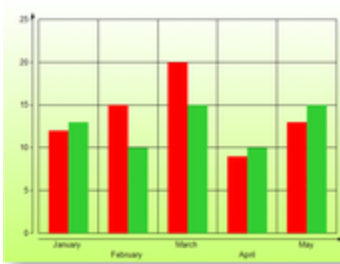


Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value |

Below is an example of how to set the custom chart properties at run time for a bar chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 50.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 50f;
```

Bezier Chart

Use a Bezier or spline chart to compare trends over a period of time or across categories. It is a line chart that plots curves through the data points in a series.

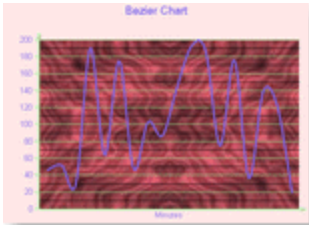


Chart Information

Chart Information

| | |
|--|----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | None |

Doughnut Chart

A doughnut chart shows how the percentage of each data item contributes to the total.



Chart Information

Chart Information

| | |
|--|--|
| Number of Y values per data point | 1 |
| Number of Series | 1 |
| Marker Support | Series or Data Point |
| Custom Properties | <p>ExplodeFactor gets or sets the amount of separation between data point values.</p> <p>HoleSize gets or sets the inner radius of the chart.</p> <p>OutsideLabels gets or sets a value indicating whether the data point labels appear outside the chart.</p> <p>StartAngle gets or sets the horizontal start angle for the series.</p> |

In order to show each section of the pie in a different color, the Background property for each data point must be set. Below is an example of how to set custom chart properties at run time for a doughnut chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("ExplodeFactor") = 0.0F
Me.ChartControl1.Series(0).Properties("HoleSize") = 0.25F
Me.ChartControl1.Series(0).Properties("OutsideLabels") = False
Me.ChartControl1.Series(0).Properties("StartAngle") = 0.0F
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
this.chartControl1.Series[0].Properties["ExplodeFactor"] = 0f;
this.chartControl1.Series[0].Properties["HoleSize"] = 0.25f;
this.chartControl1.Series[0].Properties["OutsideLabels"] = false;
this.chartControl1.Series[0].Properties["StartAngle"] = 0f;
```

Gantt Chart

The Gantt chart is a project management tool used to chart the progress of individual project tasks. The chart compares project task completion to the task schedule.

In a Gantt chart the X and Y axes are reversed. AxisX is vertical and AxisY is horizontal.

**Chart Information****Chart Information**

| | |
|--|--|
| Number of Y values per data point | 2 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value. |

Below is an example of how to set the custom chart properties at run time for a Gantt chart.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

```
Me.ChartControl1.Series(0).Properties("Gap") = 50.0F
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
this.chartControl1.Series[0].Properties["Gap"] = 50f;
```

Horizontal Bar Chart

Use a horizontal bar chart to compare values of items across categories with the axes reversed.



Chart Information

Chart Information

Number of Y values per data point 1

Number of Series 1 or more

Marker Support Series or Data Point

Custom Properties **Gap** gets or sets the space between the bars of each X axis value.

Below is an example of how to set the custom chart properties at run time for a horizontal bar chart as shown above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 65.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 65f;
```

Line Chart

Use a line chart to compare trends over a period of time or across categories.

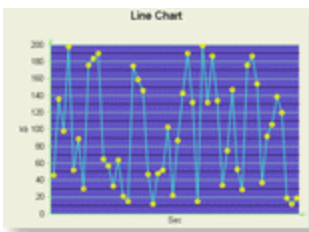


Chart Information

Chart Information

Number of Y values per data point 1

Number of Series 1 or more

Marker Support Series or Data Point

Custom Properties

None

Scatter Chart

Use a scatter chart to compare values across categories.

**Chart Information****Chart Information****Number of Y values per data point**

1

Number of Series

1 or more

Marker Support

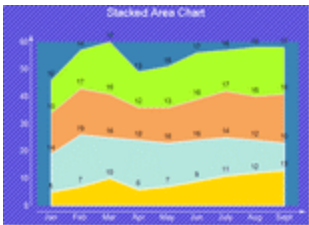
Series or Data Point

Custom Properties

None

Stacked Area Chart

A stacked area chart is an area chart with two or more data series stacked one on top of the other. Use this chart to show how each value contributes to a total.

**Chart Information****Chart Information****Number of Y values per data point**

1

Number of Series

1 or more

Marker Support

Series or Data Point

Custom Properties

None

Stacked Bar Chart

A stacked bar chart is a bar chart with two or more data series stacked one on top of the other. Use this chart to show how each value contributes to a total.

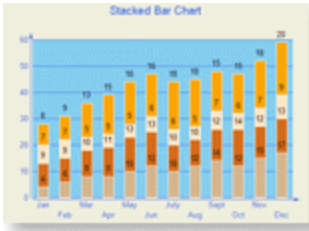


Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value |

Below is an example of how to set the custom chart properties at run time for a StackedBar chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 100f;
```

Stacked Area 100 Percent Chart

A stacked area 100% chart is an area chart with two or more data series stacked one on top of the other to sum up to 100%. Use this chart to show how each value contributes to a total with the relative size of each series representing its contribution to the total.

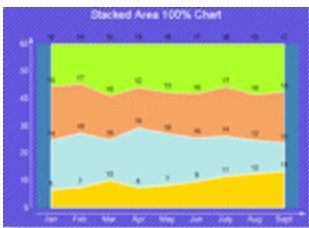


Chart Information

Chart Information

| | |
|--|----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | None |

Stacked Bar 100 Percent Chart

A StackedBar100Pct chart is a bar chart with two or more data series stacked one on top of the other to sum up to 100%. Use this chart to show how each value contributes to a total with the relative size of each series representing its contribution to the total.

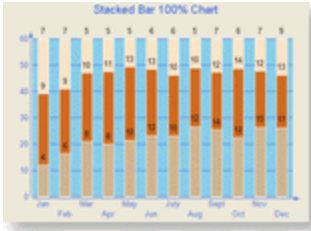


Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value |

Below is an example of how to set the custom chart properties at run time for a StackedBar100Pct chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 100f;
```

3D Charts

The ActiveReports Chart control can draw a number of 3D chart types:

- Area 3D, Bar 3D, Clustered Bar
- Doughnut 3D, Funnel 3D, Pyramid 3D
- Horizontal Bar 3D, Line 3D
- Stacked Bar 3D, Stacked Bar 3D 100 Percent

See below for details on each of the 3D chart types.

Note: To see a chart in three dimensions, set the **ProjectionType** to **Orthogonal**. The ProjectionType is found in the ChartArea Collection dialog in the Projection section.

Area 3D Chart

Use a 3D area chart to compare trends in two or more data series over a period of time or in specific categories, so that data can be viewed side by side.

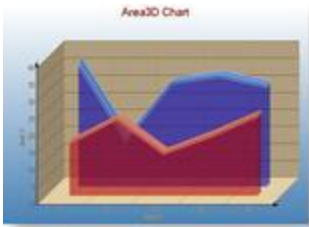


Chart Information

Number of Y values per data point

1

Number of Series

1 or more

Marker Support

Series or Data Point

Custom Properties

LineBackdrop gets or sets the backdrop information for the 3D line.

Thickness gets or sets the thickness of the 3D line.

Width gets or sets the width of the 3D line.

Chart Information

Below is an example of how to set the custom chart properties at run time for a 3D area chart as shown for the first series in the image above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("LineBackdrop") = New
Chart.Graphics.Backdrop(Color.Red, CType(150, Byte))
Me.ChartControl1.Series(0).Properties("Thickness") = 5.0F
Me.ChartControl1.Series(0).Properties("Width") = 30.0F
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["LineBackdrop"] = new
Chart.Graphics.Backdrop(Color.Red, ((System.Byte) 150));
this.chartControl1.Series[0].Properties["Thickness"] = 5f;
this.chartControl1.Series[0].Properties["Width"] = 30f;
```

Bar 3D Chart

Use a 3D bar chart to compare values of items across categories, allowing the data to be viewed conveniently in a 3D

format.



Chart Information

Chart Information

Number of Y values per data point

1

Number of Series

1 or more

Marker Support

Series or Data Point

Custom Properties

BarTopPercent gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.

BarType gets or sets the type of bars that is displayed.

Gap gets or sets the space between the bars of each X axis value.

RotationAngle gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.

VertexNumber gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Below is an example of how to set the custom chart properties at run time for a 3D bar chart as shown above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("BarTopPercent") = 80.0F
Me.ChartControl1.Series(0).Properties("BarType") = BarType.Custom
Me.ChartControl1.Series(0).Properties("Gap") = 65.0F
Me.ChartControl1.Series(0).Properties("RotationAngle") = 0.0F
Me.ChartControl1.Series(0).Properties("VertexNumber") = 6
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["BarTopPercent"] = 80f;
this.chartControl1.Series[0].Properties["BarType"] = BarType.Custom;
this.chartControl1.Series[0].Properties["Gap"] = 65f;
this.chartControl1.Series[0].Properties["RotationAngle"] = 0f;
this.chartControl1.Series[0].Properties["VertexNumber"] = 6;
```

3D Clustered Bar Chart

Use a 3D clustered bar chart to compare values of items across categories, allowing the data to be viewed conveniently in a 3D format.

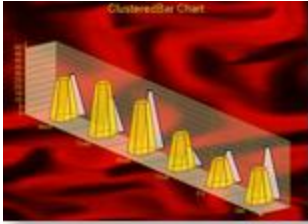


Chart Information

Chart Information

Number of Y values per data point

1

Number of Series

1 or more

Marker Support

Series or Data Point

Custom Properties

BarTopPercent gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.

BarType gets or sets the type of bars that are displayed.

Gap gets or sets the space between the bars of each X axis value.

RotationAngle gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.

VertexNumber gets or sets the number of vertices for the data point, used to create custom 3D bar shapes. Can only be used with the Custom BarType. Bars must contain 3 or more vertices.

Below is an example of how to set the custom chart properties at run time for a 3D clustered bar chart as shown above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' set the custom properties for series 1
Me.ChartControl1.Series(0).Properties("BarTopPercent") = 50.0F
Me.ChartControl1.Series(0).Properties("BarType") = BarType.Custom
Me.ChartControl1.Series(0).Properties("Gap") = 300.0F
Me.ChartControl1.Series(0).Properties("RotationAngle") = 0.0F
Me.ChartControl1.Series(0).Properties("VertexNumber") = 6

' set the custom properties for series 2
Me.ChartControl1.Series(1).Properties("BarTopPercent") = 20.0F
Me.ChartControl1.Series(1).Properties("BarType") = BarType.Custom
Me.ChartControl1.Series(1).Properties("Gap") = 300.0F
Me.ChartControl1.Series(1).Properties("RotationAngle") = 90.0F
Me.ChartControl1.Series(1).Properties("VertexNumber") = 3
```

To write the code in C#**C# code. Paste ABOVE the report class.**

```
using DataDynamics.ActiveReports.Chart;
```

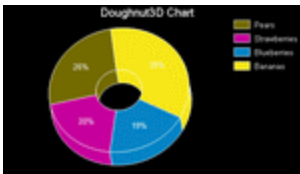
C# code. Paste INSIDE the section Format event.

```
// set the custom properties for series 1
this.chartControl1.Series[0].Properties["BarTopPercent"] = 50f;
this.chartControl1.Series[0].Properties["BarType"] = BarType.Custom;
this.chartControl1.Series[0].Properties["Gap"] = 300f;
this.chartControl1.Series[0].Properties["RotationAngle"] = 0f;
this.chartControl1.Series[0].Properties["VertexNumber"] = 6;

// set the custom properties for series 2
this.chartControl1.Series[1].Properties["BarTopPercent"] = 20f;
this.chartControl1.Series[1].Properties["BarType"] = BarType.Custom;
this.chartControl1.Series[1].Properties["Gap"] = 300f;
this.chartControl1.Series[1].Properties["RotationAngle"] = 90f;
this.chartControl1.Series[1].Properties["VertexNumber"] = 3;
```

Doughnut 3D Chart

A 3D doughnut chart shows how the percentage of each data item contributes to a total percentage, allowing the data to be viewed in a 3D format.

**Chart Information****Chart Information**

Number of Y values per data point 1

Number of Series 1

Marker Support Series or Data Point

Custom Properties

- ExplodeFactor** gets or sets the amount of separation between data point values. The value must be less than or equal to 1. To explode one section of the doughnut chart, set ExplodeFactor on the data point instead of on the series.
- HoleSize** gets or sets the inner radius of the chart. If set to 0, the chart looks like a pie chart. The value must be less than or equal to 1.
- OutsideLabels** gets or sets a value indicating whether the data point labels appear outside of the graph.
- StartAngle** gets or sets the horizontal start angle for the series data points.

Below is an example of how to set the custom chart properties at run time for a 3D doughnut chart as shown in the

image above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControll1.Series(0).Properties("ExplodeFactor") = 0.0F
Me.ChartControll1.Series(0).Properties("HoleSize") = 0.33F
Me.ChartControll1.Series(0).Properties("OutsideLabels") = False
Me.ChartControll1.Series(0).Properties("StartAngle") = 50.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControll1.Series[0].Properties["ExplodeFactor"] = 0f;
this.chartControll1.Series[0].Properties["HoleSize"] = .33f;
this.chartControll1.Series[0].Properties["OutsideLabels"] = false;
this.chartControll1.Series[0].Properties["StartAngle"] = 50f;
```

Funnel 3D Chart

A 3D funnel chart shows how the percentage of each data item contributes to the whole, allowing the data to be viewed in a three dimensional format.

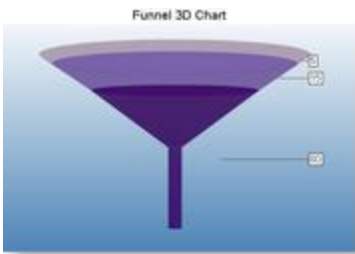


Chart Information

Number of Y values per data points 1

Number of Series 1

Marker Support Series or Data Point

Custom Properties

BaseStyle Gets or sets a circular or square base drawing style for the 3D funnel chart.

CalloutLine Gets or sets the style for a line connecting the marker label to its corresponding funnel section. The default value is a black one-point line.

FunnelStyle Gets or sets the Y value for the series points to the width or height of the funnel. The default value is YIsHeight.

MinPointHeight Gets or sets the minimum height allowed for a data point in the funnel chart. The height is measured in relative coordinates.

NeckHeight Gets or sets the neck height for the funnel chart. This property can only be used with the FunnelStyle property set to YIsHeight. The default value is 5.

NeckWidth Gets or sets the neck width for the funnel chart. This property can only be used

Chart Information

with the `FunnelStyle` property set to `YIsHeight`. The default value is 5.

OutsideLabels Gets or sets a value indicating whether the labels are placed outside of the funnel chart. The default value is `True`.

OutsideLabelsPlacement Gets or sets a value indicating whether the data point labels appear on the left or right side of the funnel. This property can only be used with the `OutsideLabels` property set to `True`.

PointGapPct Gets or sets the amount of space between the data points of the funnel chart. The `PointGapPct` is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.

RotationAngle Gets or sets the left-to-right rotation angle of the funnel. The valid values range from -180 to 180 degrees. This property is only effective with the `Projection` property set to `Orthogonal` and the `BaseStyle` property set to `SquareBase`.

Below is an example of how to set the custom chart properties at run time for a 3D funnel chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("BaseStyle") = BaseStyle.SquareBase
    .Properties("CalloutLine") = New Line(Color.Black, 2, LineStyle.Dot)
    .Properties("FunnelStyle") = FunnelStyle.YIsWidth
    .Properties("MinPointHeight") = 10.0F
    .Properties("NeckWidth") = 20.0F
    .Properties("NeckHeight") = 5.0F
    .Properties("OutsideLabels") = True
    .Properties("OutsideLabelsPlacement") = LabelsPlacement.Right
    .Properties("PointGapPct") = 3.0F
    .Properties("RotationAngle") = 3.0F
End With
```

To write the code in Visual Basic.NET

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart;
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["BaseStyle"] = BaseStyle.SquareBase;
this.chartControl1.Series[0].Properties["CalloutLine"] = new Line(Color.Black, 2,
LineStyle.Dot);
this.chartControl1.Series[0].Properties["FunnelStyle"] = FunnelStyle.YIsWidth;
this.chartControl1.Series[0].Properties["MinPointHeight"] = 10f;
this.chartControl1.Series[0].Properties["NeckWidth"] = 20f;
this.chartControl1.Series[0].Properties["NeckHeight"] = 5f;
this.chartControl1.Series[0].Properties["OutsideLabels"] = true;
this.chartControl1.Series[0].Properties["OutsideLabelsPlacement"] =
LabelsPlacement.Right;
this.chartControl1.Series[0].Properties["PointGapPct"] = 3f;
this.chartControl1.Series[0].Properties["RotationAngle"] = 3f;
```

Horizontal Bar 3D Chart

Use a horizontal 3D bar chart to compare values of items across categories, allowing the data to be viewed conveniently in a 3D format with the axes reversed.

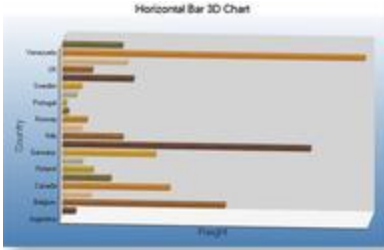


Chart Information

Chart Information

Number of Y values per data point 1

Number of Series 1 or more

Marker Support Series or Data Point

Custom Properties **Gap** gets or sets the space between the bars of each X axis value.
PointBarDepth Gets or sets the thickness of the 3D bar.

Below is an example of how to set the custom chart properties at run time for a 3D bar chart as shown above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControll1.Series(0).Properties("Gap") = 65.0F
Me.ChartControll1.Series(0).Properties("PointBarDepth") = 100
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControll1.Series[0].Properties["Gap"] = 65f;
this.chartControll1.Series[0].Properties["PointBarDepth"] = 100;
```

Line 3D Chart

Use a 3D line chart to compare trends over a period of time or in certain categories in a 3D format.

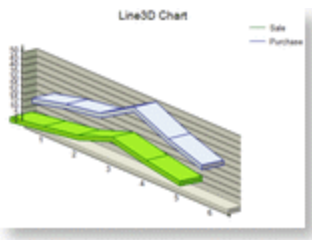


Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | <p>LineBackdrop gets or sets the backdrop information for the 3D line.</p> <p>Thickness gets or sets the thickness of the 3D line.</p> <p>Width gets or sets the width of the 3D line.</p> |

Below is an example of how to set the custom chart properties at run time for a 3D line chart as shown for the first series in the image above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("LineBackdrop") = New Backdrop(Color.GreenYellow)
Me.ChartControl1.Series(0).Properties("Thickness") = 8.0F
Me.ChartControl1.Series(0).Properties("Width") = 40.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["LineBackdrop"] = new
Backdrop(Color.GreenYellow);
this.chartControl1.Series[0].Properties["Thickness"] = 8f;
this.chartControl1.Series[0].Properties["Width"] = 40f;
```

Pyramid 3D Chart

A 3D Pyramid chart shows how the percentage of each data item contributes to the whole, allowing the data to be viewed in a three dimensional format.

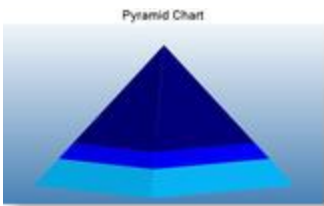


Chart Information

| | |
|--|-----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 |
| Marker Support | Series or Data Points |

Chart Information

Custom Properties

BaseStyle Gets or sets a circular or square base drawing style for the 3D pyramid chart.

CalloutLine Gets or sets the style for a line connecting the marker label to its corresponding pyramid section. The default value is a black one-point line.

MinPointHeight Gets or sets the minimum height allowed for a data point in the pyramid chart. The height is measured in relative coordinates.

OutsideLabels Gets or sets a value indicating whether the labels are placed outside of the pyramid chart. The default value is True.

OutsideLabelsPlacement Gets or sets a value indicating whether the data point labels appear on the left or right side of the pyramid. This property can only be used with the OutsideLabels property set to True.

PointGapPct Gets or sets the amount of space between the data points of the pyramid chart. The PointGapPct is measured in relative coordinates. The default value is 0, and valid values range from 0 to 100.

RotationAngle Gets or sets the left-to-right rotation angle of the pyramid. The valid values range from -180 to 180 degrees. This property is only effective with the Projection property set to Orthogonal and the BaseStyle property set to SquareBase.

Below is an example of how to set the custom chart properties at run time for a Pyramid chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControll1.Series(0)
    .Properties("BaseStyle") = BaseStyle.SquareBase
    .Properties("MinPointHeight") = 10.0F
    .Properties("OutsideLabels") = True
    .Properties("OutsideLabelsPlacement") = LabelsPlacement.Right
    .Properties("PointGapPct") = 3.0F
    .Properties("RotationAngle") = 3.0F
End With
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart;
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControll1.Series[0].Properties["BaseStyle"] = BaseStyle.SquareBase;
this.chartControll1.Series[0].Properties["MinPointHeight"] = 10f;
this.chartControll1.Series[0].Properties["OutsideLabels"] = true;
this.chartControll1.Series[0].Properties["OutsideLabelsPlacement"] =
LabelsPlacement.Right;
this.chartControll1.Series[0].Properties["PointGapPct"] = 3f;
this.chartControll1.Series[0].Properties["RotationAngle"] = 3f;
```

Stacked Bar 3D Chart

Use a 3D bar graph to compare values of items across categories, allowing the data to be viewed conveniently in a 3D format. A stacked bar graph is a bar graph with two or more data series stacked on top of each other. Use this graph to show how each value contributes to a total.

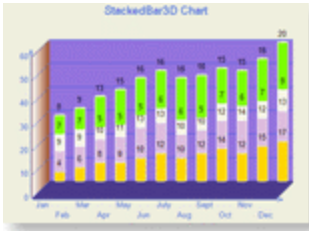


Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value |

Below is an example of how to set the custom chart properties at run time for a StackedBar3D chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 100f;
```

Stacked Bar 3D 100 Percent Chart

A StackedBar3D100Pct chart is a bar chart with two or more data series stacked one on top the other three dimensionally to sum up to 100%. Use this chart to show how each value contributes to a total with the relative size of each series representing its contribution to the total.



Chart Information

Chart Information

| | |
|--|---|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | Gap gets or sets the space between the bars of each X axis value |

Below is an example of how to set the custom chart properties at run time for a StackedBar3D100Pct chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["Gap"] = 100f;
```

XY Charts

The ActiveReports Chart control can draw a number of XY chart types:

- Bubble, Bubble XY
- Line XY
- Plot XY

See below for details on each of the XY chart types.

Bubble Chart

The Bubble chart is an XY chart in which bubbles represent data points. The first Y value is used to plot the bubble along the Y axis, and the second Y value is used to set the size of the bubble. The bubble shape can be changed using the series Shape property.

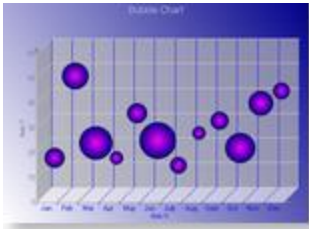


Chart Information

Number of Y values per data point

2

Number of Series

1 or more

Marker Support

Series or Data Point. Marker labels use the second Y value as the default value.

Custom Properties

MaxSizeFactor gets or sets the maximum size of the bubble radius. Values must be less than or equal to 1. Default is .25.

MaxValue gets or sets the bubble size that is used as the maximum.

MinValue gets or sets the bubble size that is used as the minimum.

Shape gets or sets the shape of the bubbles. Uses or returns a valid MarkerStyle enumeration value.

Chart Information

Below is an example of setting the custom chart properties at run time for a bubble chart as shown in the image above.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

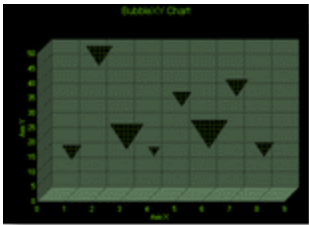
```
Me.ChartControl1.Series(0).Properties("MaxSizeFactor") = 0.25F
Me.ChartControl1.Series(0).Properties("MaxValue") = 55.0R
Me.ChartControl1.Series(0).Properties("MinValue") = 5.0R
Me.ChartControl1.Series(0).Properties("Shape") = MarkerStyle.Circle
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
this.chartControl1.Series[0].Properties["MaxSizeFactor"] = .25f;
this.chartControl1.Series[0].Properties["MaxValue"] = 55D;
this.chartControl1.Series[0].Properties["MinValue"] = 5D;
this.chartControl1.Series[0].Properties["Shape"] = MarkerStyle.Circle;
```

Bubble XY Chart

The Bubble XY chart is an XY chart in which bubbles represent data points. The BubbleXY uses a numerical X axis and plots the x values and first set of Y values on the chart. The second Y value is used to set the size of the bubble.

**Chart Information****Chart Information**

| | |
|--|---|
| Number of Y values per data point | 2 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point. Marker labels use the second Y value as the default value. |
| Custom Properties | <p>MaxSizeFactor gets or sets the maximum size of the bubble radius. Values must be less than or equal to 1. Default is .25.</p> <p>MaxValue gets or sets the bubble size that is used as the maximum.</p> <p>MinValue gets or sets the bubble size that is used as the minimum.</p> <p>Shape gets or sets the shape of the bubbles. Uses or returns a valid MarkerStyle enumeration value.</p> |

Below is an example of setting the custom chart properties at run time for a bubble XY chart as shown in the image above.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

```
Me.ChartControl1.Series(0).Properties("MaxSizeFactor") = 0.25F
Me.ChartControl1.Series(0).Properties("MaxValue") = 50.0R
Me.ChartControl1.Series(0).Properties("MinValue") = 0.0R
Me.ChartControl1.Series(0).Properties("Shape") = MarkerStyle.InvTriangle
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["MaxSizeFactor"] = .25f;
this.chartControl1.Series[0].Properties["MinValue"] = 0D;
this.chartControl1.Series[0].Properties["MaxValue"] = 50D;
this.chartControl1.Series[0].Properties["Shape"] = MarkerStyle.InvTriangle;
```

Line XY Chart

A line XY chart plots points on the X and Y axes as one series and uses a line to connect points to each other.



Chart Information

Chart Information

| | |
|--|----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | None |

Plot XY Chart

A plot XY chart shows the relationships between numeric values in two or more series sets of XY values.

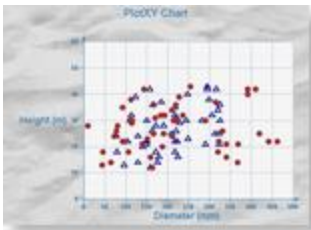


Chart Information

Chart Information

| | |
|--|----------------------|
| Number of Y values per data point | 1 |
| Number of Series | 1 or more |
| Marker Support | Series or Data Point |
| Custom Properties | None |

Financial Charts

The ActiveReports Chart control can draw a number of financial chart types:

- Candle, High Low, High Low Open Close
- Kagi, Renko
- Point and Figure, Three Line Break

See below for details on each of the financial chart types.

Candle Chart

A candle chart displays stock information using High, Low, Open and Close values. The size of the wick line is determined by the High and Low values, while the size of the bar is determined by the Open and Close values. The bar is displayed using different colors, depending on whether the price of the stock has gone up or down.

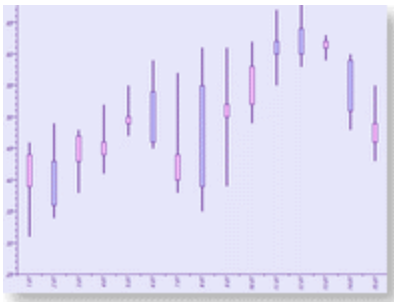


Chart Information

Chart Information

Number of Y values per data point

4 (The first value is the high figure, the second is the low figure, the third is the opening figure, and the fourth is the closing figure.)

Number of Series

1 or more

Marker Support

Series or Data Point. Marker labels use the first Y value as the default value.

Custom Properties

BodyDownswingBackdrop gets or sets the backdrop information used to fill the rectangle for data points in which the closing figure is lower than the opening figure.
BodyUpswingBackdrop gets or sets the backdrop information used to fill the rectangle for data points in which the closing figure is higher than the opening figure.
BodyWidth gets or sets the width of the rectangle used to show upswing or downswing.
WickLine gets or sets the line information for the wick line.

Below is an example of how to set the custom chart properties at run time for a candle chart as shown in the image above.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New
```

```

Chart.Graphics.Backdrop(Color.FromArgb(255, 192, 255))
    .Properties("BodyUpswingBackdrop") = New Chart.Graphics.Backdrop(Color.FromArgb(192,
192, 255))
    .Properties("WickLine") = New Chart.Graphics.Line(Color.Indigo)
    .Properties("BodyWidth") = 7.0F
End With

```

To write the code in C#

C# code. Paste ABOVE the report class.

```
Using DataDynamics.ActiveReports.Chart.Graphics
```

C# code. Paste INSIDE the section Format event.

```

this.chartControll1.Series[0].Properties["BodyDownswingBackdrop"] = new
Chart.Graphics.Backdrop
    (Color.FromArgb(255, 192, 255));
this.chartControll1.Series[0].Properties["BodyUpswingBackdrop"] = new
Chart.Graphics.Backdrop
    (Color.FromArgb(192, 192, 255));
this.chartControll1.Series(0).Properties("WickLine") = new
Chart.Graphics.Line(Color.Indigo);
this.chartControll1.Series[0].Properties["BodyWidth"] = 7f;

```

HiLo Chart

A HiLo chart displays stock information using High and Low or Open and Close values. The length of the HiLo line is determined by the High and Low values or the Open and Close values.

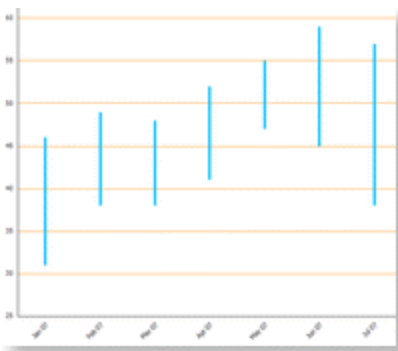


Chart Information

Number of Y values per data point

2

Number of Series

1 or more

Marker Support

Series or Data Point. Marker labels use the first Y value as the default value.

Custom Properties

HiLoLine gets or sets the line information for the HiLo line.

Below is an example of how to set the custom chart properties at run time for a HiLo chart as shown in the image above.

Chart Information

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

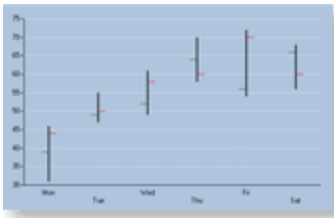
```
Me.ChartControll1.Series(0).Properties("HiloLine") = New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.DeepSkyBlue, 4)
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
this.chartControll1.Series[0].Properties["HiloLine"] = new
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.DeepSkyBlue, 4);
```

HiLo OpenClose Chart

A HiLo OpenClose chart displays stock information using High, Low, Open and Close values. Opening values are displayed using lines to the left, while lines to the right indicate closing values.

**Chart Information****Number of Y values per data point**

4

Number of Series

1 or more

Marker Support

Series or Data Point. Marker labels use the first Y value as the default value.

Custom Properties

HiloLine gets or sets the line information for the HiLo line.
CloseLine gets or sets the line information for the close line.
OpenLine gets or sets the line information for the open line.
TickLen gets or sets the length of the tick for the open and close lines.

Below is an example of how to set the custom chart properties at run time for a HiLoOpenClose chart as shown in the image above.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste ABOVE the report class.**

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControll1.Series(0)
    .Properties("OpenLine") = New Chart.Graphics.Line(Color.Green)
    .Properties("CloseLine") = New Chart.Graphics.Line(Color.Red)
    .Properties("HiloLine") = New Chart.Graphics.Line(Color.Black, 2)
    .Properties("TickLen") = 10.0F
```

End With

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControll1.Series[0].Properties["OpenLine"] = new
Chart.Graphics.Line(Color.Green);
this.chartControll1.Series[0].Properties["CloseLine"] = new
Chart.Graphics.Line(Color.Red);
this.chartControll1.Series[0].Properties["HiloLine"] = new
Chart.Graphics.Line(Color.Black, 2);
this.chartControll1.Series[0].Properties["TickLen"] = 10f;
```

Kagi Chart

A Kagi chart displays supply and demand trends using a sequence of linked vertical lines. The thickness and direction of the lines vary depending on the price movement. If closing prices go in the direction of the previous Kagi line, then that Kagi line is extended. However, if the closing price reverses by the preset reversal amount, a new Kagi line is charted in the next column in the opposite direction. Thin lines indicate that the price breaks the previous low (supply) while thick lines indicate that the price breaks the previous high (demand).

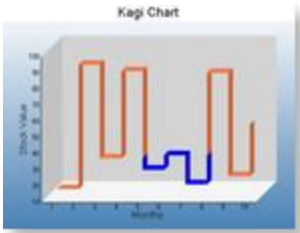


Chart Information

Chart Information

Number of Y values per data point 1

Number of Series 1

Marker Support Series or Data Points

Custom Properties

- BodyDownswingBackdrop** Gets or sets the style and color settings for the three-dimensional side view of downswing Kagi lines. This property is only available with the Kagi 3D chart type, and is only effective when the Width property is set to a value higher than 25.
- BodyUpswingBackdrop** Gets or sets the style and color settings for the three-dimensional side view of upswing Kagi lines. This property is only available with the Kagi 3D chart type, and is only effective when the Width property is set to a value higher than 25.
- DownswingLine** Gets or sets the style and color settings to use for a Kagi line which charts a

price decrease.

ReversalAmount Gets or sets the amount that a price must shift in order for the Kagi line to change direction.

UpswingLine Gets or sets the style and color settings to use for a Kagi line which charts a price increase.

Width Gets or sets the width of the three-dimensional side view of the Kagi lines. This property is only available with the Kagi 3D chart type, and must be set higher than its default value of 1 in order to display body downswing and upswing backdrops.

Below is an example of how to set the custom chart properties at run time for a Kagi chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New Backdrop(Color.Red)
    .Properties("BodyUpswingBackdrop") = New Backdrop(Color.Blue)
    .Properties("DownswingLine") = New Chart.Graphics.Line(Color.DarkRed)
    .Properties("ReversalAmount") = "25"
    .Properties("UpswingLine") = New Chart.Graphics.Line(Color.DarkBlue)
    .Properties("Width") = 50.0F
End With
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["BodyDownswingBackdrop"] = new
Backdrop(Color.Red);
this.chartControl1.Series[0].Properties["BodyUpswingBackdrop"] = new
Backdrop(Color.Blue);
this.chartControl1.Series[0].Properties["DownswingLine"] = new
Chart.Graphics.Line(Color.DarkRed);
this.chartControl1.Series[0].Properties["ReversalAmount"] = "25";
this.chartControl1.Series[0].Properties["UpswingLine"] = new
Chart.Graphics.Line(Color.DarkBlue);
this.chartControl1.Series[0].Properties["Width"] = 50f;
```

Point and Figure Chart

The point and figure chart uses stacked columns of X's to indicate that demand exceeds supply and columns of O's to indicate that supply exceeds demand to define pricing trends. A new X or O is added to the chart if the price moves higher or lower than the BoxSize value. A new column is added when the price reverses to the level of the BoxSize value multiplied by the ReversalAmount. The use of these values in the point and figure chart to calculate pricing trends makes this chart best suited for long-term financial analysis.

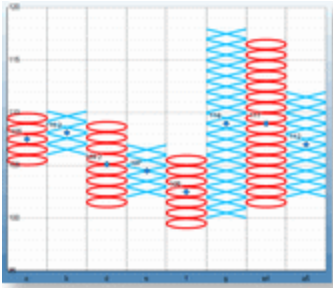


Chart Information

Chart Information

| | |
|--|--|
| Number of Y values per data point | 2 |
| Number of Series | 1 |
| Marker Support | Series or Data Points |
| Custom Properties | <p>BoxSize Gets or sets the amount a price must change in order to create another X or O.</p> <p>DownswingLine Gets or sets the style and color settings for the downswing O's.</p> <p>ReversalAmount Gets or sets the amount that a price must shift in order for a new column to be added.</p> <p>UpswingLine Gets or sets the style and color settings for the upswing X's.</p> |

Below is an example of how to set the custom chart properties at run time for a Point and Figure chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("DownswingLine") = New Chart.Graphics.Line(Color.Red)
    .Properties("UpswingLine") = New Chart.Graphics.Line(Color.Blue)
    .Properties("BoxSize") = 3.0F
End With
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["DownswingLine"] = new
Chart.Graphics.Line(Color.Red);
this.chartControl1.Series[0].Properties["UpswingLine"] = new
Chart.Graphics.Line(Color.Blue);
this.chartControl1.Series[0].Properties["BoxSize"] = 3f;
```

Renko Chart

The Renko chart uses bricks of uniform size to chart price movement. When a price moves to a greater or lesser value than the preset `BoxSize` value required to draw a new brick, a new brick is drawn in the succeeding column. The change in box color and direction signifies a trend reversal.

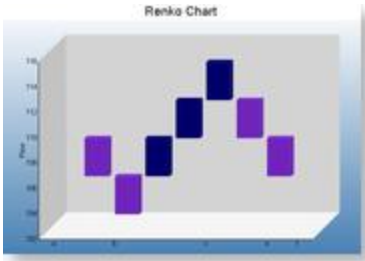


Chart Information

Number of Y values per data point

2

Number of Series

1

Marker Support

Series or Data Points

Custom Properties

BodyDownswingBackdrop Gets or sets the style and color settings for the downswing bricks.

BodyUpswingBackdrop Gets or sets the style and color settings for the upswing bricks.

BoxSize Gets or sets the amount a price must change in order to create another brick.

Chart Information

Below is an example of how to set the custom chart properties at run time for a Renko chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New Backdrop(Color.BlueViolet)
    .Properties("BodyUpswingBackdrop") = New Backdrop(Color.Navy)
    .Properties("BoxSize") = 3.0F
End With
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["BodyDownswingBackdrop"] = new
Backdrop(Color.BlueViolet);
this.chartControl1.Series[0].Properties["BodyUpswingBackdrop"] = new
```

```
Backdrop(Color.Navy);
this.chartControl1.Series[0].Properties["BoxSize"] = 3f;
```

Three Line Break Chart

A Three Line Break chart uses vertical boxes or lines to illustrate price changes of an asset or market. Movements are depicted with box colors and styles; movements that continue the trend of the previous box paint similarly while movements that trend oppositely are indicated with a different color and/or style. The opposite trend is only drawn if its value exceeds the extreme value of the previous three boxes or lines. The below Three Line Break depicts upward pricing movement with black boxes and downward pricing movement with red boxes.

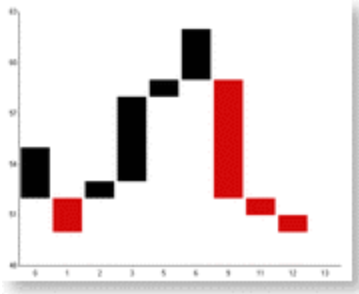


Chart Information

Chart Information

| | |
|--|--|
| Number of Y values per data point | 1 |
| Number of Series | 1 |
| Marker Support | Series or Data Points |
| Chart-Specific Properties | <p>BodyDownswingBackdrop Gets or sets the style and color settings for the downswing boxes.</p> <p>BodyUpswingBackdrop Gets or sets the style and color settings for the upswing boxes.</p> <p>NewLineBreak Gets or sets the number of previous boxes/lines that must be compared before a new box/line is drawn. The default value is 3.</p> |

Below is an example of how to set the custom chart properties at run time for a Three Line Break chart.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New Backdrop(Color.Red)
    .Properties("BodyUpswingBackdrop") = New Backdrop(Color.Black)
    .Properties("NewLineBreak") = 3
End With
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics;
```

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Series[0].Properties["BodyDownswingBackdrop"] = new  
Backdrop(Color.Red);  
this.chartControl1.Series[0].Properties["BodyUpswingBackdrop"] = new  
Backdrop(Color.Black);  
this.chartControl1.Series[0].Properties["NewLineBreak"] = 3;
```

Chart Appearance

These topics introduce you to some basic information about manipulating the appearance of Charts.

Chart Effects

Learn about the visual effects possible with the Chart control.

Chart Control Items

Learn about Chart control items that can be used to customize your chart.

Chart Axes and Walls

Learn some basic information about Chart axes and walls.

Chart Effects

Colors

In the Chart control, colors can be used in different ways to enhance the chart's appearance, distinguish different series, point out or draw attention to data information such as averages, and more.

Color Palettes

The Chart control includes several pre-defined color palettes that can be used to automatically set the colors for data values in a series. The pre-defined palettes are as follows:

- **Cascade** (default) A cascade of eight cool colors ranging from deep teal down through pale orchid.
- **Confetti** A sprinkling of bright and pastel colors.
- **Iceberg** A range of the soft blues and greys found in an iceberg.
- **Springtime** The colors of spring, in deep green, two vivid colors and five pastels.
- **None** All data is drawn using the same teal color.

These enumerated values are accessed through the Series class with code like the following.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to set the color palette for the series.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

```
Me.ChartControl1.Series(0).ColorPalette =  
DataDynamics.ActiveReports.Chart.ColorPalette.Iceburg
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
this.chartControll.Series[0].ColorPalette =  
DataDynamics.ActiveReports.Chart.ColorPalette.Iceburg;
```

Gradients

Gradients can be used in object backdrops to enhance the visual appearance of various chart items. Gradients can be used in the following chart sections:

- Chart backdrop
- Chart area backdrops
- Wall backdrops
- Title backdrops
- Legend backdrops
- Legend item backdrops (for custom legend items)
- WallRange backdrops
- Series backdrops
- Data point backdrops
- Marker backdrops
- Marker label backdrops
- Annotation TextBar backdrops

You can set gradients for a backdrop at run time by creating a BackdropItem, setting its Style property to Gradient, setting the GradientType, and setting the two colors to use for the gradient as shown in the following example.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to set gradients for the backdrop.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the report class.

```
Imports DataDynamics.ActiveReports.Chart.Graphics
```

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Dim bItem As New Chart.BackdropItem(GradientType.Vertical, Color.Purple, Color.White)  
Me.ChartControll.Backdrop = bItem
```

To write the code in C#

C# code. Paste ABOVE the report class.

```
using DataDynamics.ActiveReports.Chart.Graphics
```

C# code. Paste INSIDE the section Format event.

```
DataDynamics.ActiveReports.Chart.BackdropItem bItem = new DataDynamics.ActiveReports.  
Chart.BackdropItem(GradientType.Vertical, Color.Purple, Color.White);  
this.chartControll.Backdrop = bItem;
```

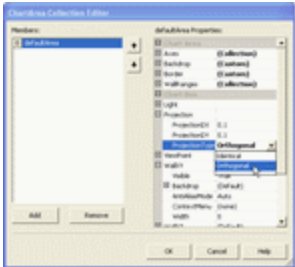
3D Effects

Using the projection and viewpoint settings, you can display your 3D chart at any angle to provide the desired view or call attention to a specific chart section.

Projection

Determine the projection for a 3D chart using three factors: the Z depth ratio, the projection type, and the projection DX and DY values.

- **ZDepth ratio** The Z depth ratio is the level of depth the Z axis has in the chart. Values range from 0 (for a 2D chart) to 1.0.
- **ProjectionType** The type of projection used for the chart. In order to show charts three dimensionally, the ProjectionType in the ChartArea Collection editor must be set to Orthogonal. To access this dialog box, click the ellipsis button next to the ChartAreas (Collection) property in the Properties Window.



- **ProjectionDX** The origin position of the Z axis in relation to the X axis. This property is valid only when the ProjectionType is Orthogonal.
- **ProjectionDY** The origin position of the Z axis in relation to the Y axis. This property is valid only when the ProjectionType is Orthogonal.
- **HorizontalRotation** The HorizontalRotation property allows you to set the degree (-90° to 90°) of horizontal rotation from which the chart is seen.
- **VerticalRotation** The VerticalRotation property allows you to set the degree (-90° to 90°) of vertical rotation from which the chart is seen.

Alpha Blending

The Backdrop class in the Chart control has an Alpha property which employs GDI+, and is used to set the transparency level of each object's backdrop. GDI+ uses 32 bits overall and 8 bits per alpha, red, green, and blue channels respectively to indicate the transparency and color of an object. Like a color channel's levels of color, the alpha channel represents 256 levels of transparency.

The default value of the Alpha property is 255, which represents a fully opaque color. For a fully transparent color, set this value to 0. To blend the color of the object's backdrop with the background color, use a setting between 0 and 255.

In the Chart control, you can use the Color.FromArgb method to set the alpha and color levels for a particular chart element. The following example shows how you can use the method to set the alpha and color values for the chart backdrop.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to set the transparency of the chart objects.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Me.ChartControl1.Backdrop = New
DataDynamics.ActiveReports.Chart.BackdropItem(Color.FromArgb(100, 0, 11, 220))
```

To write the code in C#

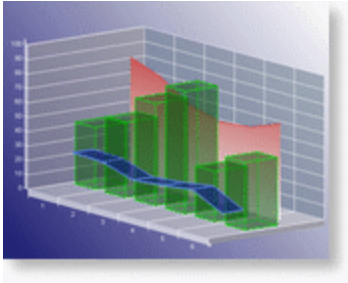
C# code. Paste INSIDE the section Format event.

```
this.chartControl1.Backdrop = new
DataDynamics.ActiveReports.Chart.BackdropItem(Color.FromArgb(100, 0, 11, 220));
```

Changing the alpha level of a chart element reveals other items that are beneath the object. Because you can set the alpha level for any chart element that supports color, you can create custom effects for any chart. For example, you can use alpha blending to combine background images with a semi-transparent chart backdrop to create a watermark look.

Lighting

The Chart control allows you to completely customize lighting options for 3D charts.



Directional Light Ratio

Using the DirectionalLightRatio property, you can control the directional or ambient intensity ratio.

Light Type

By setting the Type property to one of the enumerated LightType values, you can control the type of lighting used in the chart. The settings are as follows:

- **Ambient** An ambient light source is used. It is equal to DirectionalLightRatio = 0.
- **InfiniteDirectional** An infinite directional light source (like the sun) is used.
- **FiniteDirectional** A point light source is used.

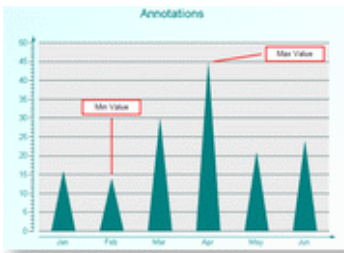
Light Source

You can also set the Source property to a Point3d object, which controls the location of the light source.

Chart Control Items

Annotations

The Chart control offers a built-in annotation tool to allow you to include floating text bars or images in your charts or call attention to specific items or values in your charts using the line and text bar controls included in the Annotation Collection Editor.



The following properties are important when setting up annotations for your chart:

- **StartPoint** Sets the starting point (X and Y axis values) for an annotation line.
- **EndPoint** Sets the end point (X and Y axis values) for an annotation line.
- **AnchorPlacement** Sets the position of the anchor point for the text bar on the chart surface.

- **AnchorPoint** Sets the point (X and Y axis values) where the text bar will be anchored based on the anchor placement selected.

The following code demonstrates creating annotation lines and text bars, setting their properties, and adding them to the series annotations collection at run time. The results are shown in the screen shot above.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create annotation lines and text bars.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the annotation lines and text bars
Dim aLine1 As New DataDynamics.ActiveReports.Chart.Annotations.AnnotationLine
Dim aLine2 As New DataDynamics.ActiveReports.Chart.Annotations.AnnotationLine
Dim aText1 As New DataDynamics.ActiveReports.Chart.Annotations.AnnotationTextBar
Dim aText2 As New DataDynamics.ActiveReports.Chart.Annotations.AnnotationTextBar

' set the properties for each line and text bar
With aLine1
    .EndPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 30.0F)
    .Line = New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red, 2)
    .StartPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 15.0F)
End With
With aLine2
    .EndPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(4.6F, 47.0F)
    .Line = New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red, 2)
    .StartPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(3.6F, 45.0F)
End With
With aText1
    .AnchorPlacement =
DataDynamics.ActiveReports.Chart.Annotations.AnchorPlacementType.Bottom
    .AnchorPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 31.0F)
    .Height = 25.0F
    .Line = New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red, 2)
    .Text = "Min Value"
    .Width = 100.0F
End With
With aText2
    .AnchorPlacement =
DataDynamics.ActiveReports.Chart.Annotations.AnchorPlacementType.Left
    .AnchorPoint = New DataDynamics.ActiveReports.Chart.Graphics.Point2d(4.7F, 47.0F)
    .Height = 25.0F
    .Line = New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red, 2)
    .Text = "Max Value"
    .Width = 100.0F
End With

' add the annotation lines and text bars to the annotations collection for the series
Me.ChartControll1.Series(0).Annotations.AddRange(New
DataDynamics.ActiveReports.Chart.Annotations.Annotation() {aLine1, aLine2, aText1,
aText2})
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

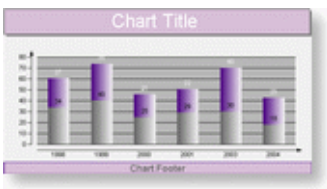
```
// create the annotation lines and text bars
DataDynamics.ActiveReports.Chart.Annotations.AnnotationLine aLine1 = new
DataDynamics.SharpGraph.Windows.Annotations.AnnotationLine();
DataDynamics.ActiveReports.Chart.Annotations.AnnotationLine aLine2 = new
DataDynamics.SharpGraph.Windows.Annotations.AnnotationLine();
DataDynamics.ActiveReports.Chart.Annotations.AnnotationTextBar aText1 = new
DataDynamics.SharpGraph.Windows.Annotations.AnnotationTextBar();
DataDynamics.ActiveReports.Chart.Annotations.AnnotationTextBar aText2 = new
DataDynamics.SharpGraph.Windows.Annotations.AnnotationTextBar();

// set the properties for each line and text bar
aLine1.EndPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 30F);
aLine1.Line = new DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red,
2);
aLine1.StartPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 15F);
aLine2.EndPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(4.6F, 47F);
aLine2.Line = new DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red,
2);
aLine2.StartPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(3.6F, 45F);
aText1.AnchorPlacement =
DataDynamics.ActiveReports.Chart.Annotations.AnchorPlacementType.Bottom;
aText1.AnchorPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(1.5F, 31F);
aText1.Height = 25F;
aText1.Line = new DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red,
2);
aText1.Text = "Min Value";
aText1.Width = 100F;
aText2.AnchorPlacement =
DataDynamics.ActiveReports.Chart.Annotations.AnchorPlacementType.Left;
aText2.AnchorPoint = new DataDynamics.ActiveReports.Chart.Graphics.Point2d(4.7F, 47F);
aText2.Height = 25F;
aText2.Line = new DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Red,
2);
aText2.Text = "Max Value";
aText2.Width = 100F;

// add the annotation lines and text bars to the annotations collection for the series
this.chartControl1.Series[0].Annotations.AddRange(new
DataDynamics.ActiveReports.Chart.Annotations.Annotation[] {aLine1, aLine2, aText1,
aText2});
```

Titles and Footers

The Chart control allows you to add custom titles to your charts. The Titles collection is accessible from the Chart object. With the ability to add as many titles as needed, dock them to any side of a chart area, change all of the font properties, add borders and shadows, make the background look the way you want it, and change the location of the text, you can easily make your titles look the way you want them to look.



The following code demonstrates creating header and footer titles, setting their properties, and adding them to the titles collection at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event

handling method for the section.

2. Add code to the handler to create header and footer titles.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the header and footer titles
Dim tHeader As New DataDynamics.ActiveReports.Chart.Title
Dim tFooter As New DataDynamics.ActiveReports.Chart.Title

' set the properties for the header
tHeader.Alignment = Chart.Alignment.Center
tHeader.Backdrop = New
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Thistle)
tHeader.Border = New DataDynamics.ActiveReports.Chart.Border(New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.DimGray), 3)
tHeader.DockArea = Me.ChartControll1.ChartAreas(0)
tHeader.Docking = Chart.DockType.Top
tHeader.Font = New DataDynamics.ActiveReports.Chart.FontInfo(System.Drawing.Color.White,
New System.Drawing.Font("Arial", 25.0F))
tHeader.Text = "Chart Title"
tHeader.Visible = True

' set the properties for the footer
tFooter.Alignment = Chart.Alignment.Center
tFooter.Backdrop = New
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Thistle)
tFooter.Border = New DataDynamics.ActiveReports.Chart.Border(New
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Indigo), 0,
System.Drawing.Color.Black)
tFooter.DockArea = Me.ChartControll1.ChartAreas(0)
tFooter.Docking = Chart.DockType.Bottom
tFooter.Font = New DataDynamics.ActiveReports.Chart.FontInfo(System.Drawing.Color.DimGray,
New System.Drawing.Font("Arial", 12.0F, System.Drawing.FontStyle.Bold))
tFooter.Text = "Chart Footer"
tFooter.Visible = True

' add the header and footer titles to the titles collection
Me.ChartControll1.Titles.AddRange(New DataDynamics.ActiveReports.Chart.Title() {tHeader,
tFooter})
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the header and footer titles
DataDynamics.ActiveReports.Chart.Title tHeader = new
DataDynamics.ActiveReports.Chart.Title();
DataDynamics.ActiveReports.Chart.Title tFooter = new
DataDynamics.ActiveReports.Chart.Title();

// set the properties for the header
tHeader.Alignment = Chart.Alignment.Center;
tHeader.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Thistle);
tHeader.Border = new DataDynamics.ActiveReports.Chart.Border(new
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.DimGray), 3);
tHeader.DockArea = this.ChartControll1.ChartAreas[0];
tHeader.Docking = Chart.DockType.Top;
tHeader.Font = new DataDynamics.ActiveReports.Chart.FontInfo(System.Drawing.Color.White,
```

```

new System.Drawing.Font("Arial", 25F));
tHeader.Text = "Chart Title";
tHeader.Visible = true;

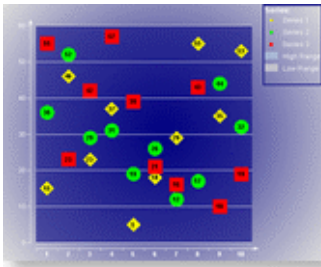
// set the properties for the footer
tFooter.Alignment = Chart.Alignment.Center;
tFooter.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Thistle);
tFooter.Border = new DataDynamics.ActiveReports.Chart.Border(new
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Indigo), 0,
System.Drawing.Color.Black);
tFooter.DockArea = this.ChartControll1.ChartAreas[0];
tFooter.Docking = Chart.DockType.Bottom;
tFooter.Font = new DataDynamics.ActiveReports.Chart.FontInfo(System.Drawing.Color.DimGray,
new System.Drawing.Font("Arial", 12F, System.Drawing.FontStyle.Bold));
tFooter.Text = "Chart Footer";
tFooter.Visible = true;

// add the header and footer titles to the titles collection
this.chartControll1.Titles.AddRange(new DataDynamics.ActiveReports.Chart.Title[]
{tHeader,tFooter});

```

Legends

The Chart control automatically creates a legend item for each series added to a chart at design time and sets the Legend property for each series by default. However, the legend's Visible property must be set to True for the legend to show with the chart. The text for each default legend entry is taken from the Name property on the series.



The following code demonstrates how to create a legend at run time, add it to the legends collection for the Chart object and set the legend property of the series to the new legend, resulting in the legend shown above.

Note: Each Series to be shown in the Legend must have a Name. If the Name property is not set, the Series does not show up in the Legend.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a legend.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```

' create the legend and title for the legend
Dim legend1 As New DataDynamics.ActiveReports.Chart.Legend
Dim lHeader As New DataDynamics.ActiveReports.Chart.Title
' set the properties for the legend title
lHeader.Backdrop = New
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(Chart.Graphics.BackdropStyle.Transparent,
Color.White, Color.White, Chart.Graphics.GradientType.Vertical,
Drawing2D.HatchStyle.DottedGrid, Nothing, Chart.Graphics.PicturePutStyle.Stretched)
lHeader.Border = New DataDynamics.ActiveReports.Chart.Border(New

```

```

Chart.Graphics.Line(Color.White, 2, Chart.Graphics.LineStyle.None), 0, Color.Black)
lHeader.Font = New DataDynamics.ActiveReports.Chart.FontInfo(Color.White, New
System.Drawing.Font("Arial", 10.0F, FontStyle.Bold))
lHeader.Text = "Series:"
' set the properties for the legend and add it to the legends collection
legend1.Alignment = DataDynamics.ActiveReports.Chart.Alignment.TopRight
legend1.Backdrop = New
DataDynamics.ActiveReports.Chart.BackdropItem(Chart.Graphics.BackdropStyle.Transparent,
Color.Gray, Color.White, Chart.Graphics.GradientType.Vertical,
Drawing2D.HatchStyle.DottedGrid, Nothing, Chart.Graphics.PicturePutStyle.Stretched)
legend1.Border = New DataDynamics.ActiveReports.Chart.Border(New
Chart.Graphics.Line(Color.Navy, 2), 0, Color.Black)
legend1.DockArea = Me.ChartControll1.ChartAreas(0)
legend1.LabelsFont = New DataDynamics.ActiveReports.Chart.FontInfo(Color.White, New System
.Drawing.Font("Arial", 9.0F))
legend1.Header = lHeader
legend1.MarginX = 5
legend1.MarginY = 5
Me.ChartControll1.Legends.Add(legend1)
' set the legend property of the series to the legend you created
Me.ChartControll1.Series(0).Legend = legend1
Me.ChartControll1.Series(1).Legend = legend1
Me.ChartControll1.Series(2).Legend = legend1

```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```

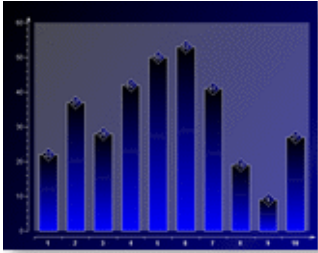
// create the legend and title for the legend
DataDynamics.ActiveReports.Chart.Legend legend1 = new
DataDynamics.ActiveReports.Chart.Legend();
DataDynamics.ActiveReports.Chart.Title lHeader = new
DataDynamics.ActiveReports.Chart.Title();
// set the properties for the legend title
lHeader.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(Chart.Graphics.BackdropStyle.Transparent,
Color.White, Color.White, Chart.Graphics.GradientType.Vertical,
System.Drawing.Drawing2D.HatchStyle.DottedGrid, null,
Chart.Graphics.PicturePutStyle.Stretched);
lHeader.Border = new DataDynamics.ActiveReports.Chart.Border(new
Chart.Graphics.Line(Color.White, 2, Chart.Graphics.LineStyle.None), 0, Color.Black);
lHeader.Font = new DataDynamics.ActiveReports.Chart.FontInfo(Color.White, new Font("Arial",
10F, FontStyle.Bold));
lHeader.Text = "Series:";
// set the properties for the legend and add it to the legends collection
legend1.Alignment = DataDynamics.ActiveReports.Chart.Alignment.TopRight;
legend1.Backdrop = new
DataDynamics.ActiveReports.Chart.BackdropItem(Chart.Graphics.BackdropStyle.Transparent,
Color.Gray, Color.White, Chart.Graphics.GradientType.Vertical,
System.Drawing.Drawing2D.HatchStyle.DottedGrid, null,
Chart.Graphics.PicturePutStyle.Stretched);
legend1.Border = new DataDynamics.ActiveReports.Chart.Border(new
Chart.Graphics.Line(Color.Navy, 2), 0, Color.Black);
legend1.DockArea = this.sharpGraph1.ChartAreas[0];
legend1.LabelsFont = new DataDynamics.ActiveReports.Chart.FontInfo(Color.White, new
Font("Arial", 9F));
legend1.Header = lHeader;
legend1.MarginX = 5;
legend1.MarginY = 5;
this.chartControll1.Legends.Add(legend1);

```

```
// set the legend property of the series to the legend you created
this.chartControl1.Series[0].Legend = legend1;
this.chartControl1.Series[1].Legend = legend1;
this.chartControl1.Series[2].Legend = legend1;
```

Markers

Use markers to show specific data series values in a chart.



The following code demonstrates how to create a marker object at run time and assign it to the Marker property of the series. The results are shown in the image above.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a marker object and assign it to the series.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the marker object
Dim marker1 As New DataDynamics.ActiveReports.Chart.Marker

' set the marker properties
marker1.Backdrop = New Chart.Graphics.Backdrop(Chart.Graphics.GradientType.Horizontal,
Color.Navy, Color.Black)
marker1.Line = New Chart.Graphics.Line(Color.White)
marker1.Label = New Chart.LabelInfo(New Chart.Graphics.Line(Color.Transparent, 0,
Chart.Graphics.LineStyle.None), New
Chart.Graphics.Backdrop(Chart.Graphics.BackdropStyle.Transparent, Color.White,
Color.White, Chart.Graphics.GradientType.Vertical,
System.Drawing.Drawing2D.HatchStyle.DottedGrid, Nothing,
Chart.Graphics.PicturePutStyle.Stretched), New Chart.FontInfo(Color.White, New
Font("Arial", 8.0F)), "{Value}", Chart.Alignment.Center)
marker1.Size = 24
marker1.Style = Chart.MarkerStyle.Diamond

' assign the marker to the series Marker property
Me.ChartControl1.Series(0).Marker = marker1
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the marker object
DataDynamics.ActiveReports.Chart.Marker marker1 = new
DataDynamics.ActiveReports.Chart.Marker();

// set the marker properties
marker1.Backdrop = new Chart.Graphics.Backdrop(Chart.Graphics.GradientType.Horizontal,
```

```

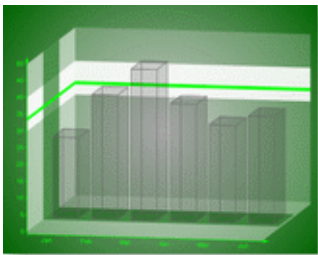
Color.Navy, Color.Black);
marker1.Line = new Chart.Graphics.Line(Color.White);
marker1.Label = new Chart.LabelInfo(new Chart.Graphics.Line(Color.Transparent, 0,
Chart.Graphics.LineStyle.None), new
Chart.Graphics.Backdrop(Chart.Graphics.BackdropStyle.Transparent, Color.White,
Color.White, Chart.Graphics.GradientType.Vertical,
System.Drawing.Drawing2D.HatchStyle.DottedGrid, null,
Chart.Graphics.PicturePutStyle.Stretched), new Chart.FontInfo(Color.White, new
Font("Arial", 8F)), "{Value}", Chart.Alignment.Center);
marker1.Size = 24;
marker1.Style = Chart.MarkerStyle.Diamond;

// assign the marker to the series Marker property
this.chartControl1.Series[0].Marker = marker1;

```

Constant Lines and Stripes

The Chart control supports constant lines and stripes through the use of the WallRanges collection. It allows you to display horizontal or vertical lines or stripes in a chart to highlight certain areas. For example, you could draw a stripe in a chart to draw attention to a high level in the data or draw a line to show the average value of the data presented.



Important properties

- **EndValue** Sets the end value on the primary axis for the wall range.
- **StartValue** Sets the start value on the primary axis for the wall range.
- **PrimaryAxis** Sets the axis on which the wall range appears.

The following code demonstrates how to create wall ranges, set their properties, and assign them to a chart area at run time. The results are shown in the image above.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create wall ranges and assign them to a chart area.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```

' create the wall range objects
Dim wallRange1 As New DataDynamics.ActiveReports.Chart.WallRange
Dim wallRange2 As New DataDynamics.ActiveReports.Chart.WallRange
Dim wallRange3 As New DataDynamics.ActiveReports.Chart.WallRange

' set the wall range properties
With wallRange1
    .Backdrop = New DataDynamics.ActiveReports.Chart.Graphics.Backdrop(Color.White)
    .Border = New DataDynamics.ActiveReports.Chart.Border(New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, Color.Black)
    .EndValue = 40

```

```

        .PrimaryAxis = ((CType(Me.ChartControl1.ChartAreas(0).Axes("AxisY"),
DataDynamics.ActiveReports.Chart.Axis)))
        .StartValue = 30
    End With
    With wallRange2
        .Backdrop = New DataDynamics.ActiveReports.Chart.Graphics.Backdrop(Color.Lime)
        .Border = New DataDynamics.ActiveReports.Chart.Border(New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, Color.Black)
        .EndValue = 34
        .PrimaryAxis = ((CType(Me.ChartControl1.ChartAreas(0).Axes("AxisY"),
DataDynamics.ActiveReports.Chart.Axis)))
        .StartValue = 33
    End With
    With wallRange3
        .Backdrop = New DataDynamics.ActiveReports.Chart.Graphics.Backdrop(Color.DarkGreen,
CType(150, Byte))
        .Border = New DataDynamics.ActiveReports.Chart.Border(New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, Color.Black)
        .EndValue = 40
        .PrimaryAxis = ((CType(Me.ChartControl1.ChartAreas(0).Axes("AxisZ"),
DataDynamics.ActiveReports.Chart.Axis)))
        .StartValue = 20
    End With

' add the wall ranges to the chart area and set wall and Z axis properties to show lines
With ChartControl1.ChartAreas(0)
    .WallRanges.AddRange(New DataDynamics.ActiveReports.Chart.WallRange() {wallRange1,
wallRange2, wallRange3})
    .WallXY.Backdrop.Alpha = 100
    .WallXZ.Backdrop.Alpha = 100
    .WallYZ.Backdrop.Alpha = 100
    .Axes(4).MajorTick.Step = 20
    .Axes(4).Max = 60
    .Axes(4).Min = 0
    .Axes(4).Visible = True
End With

```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```

// create the wall range objects
DataDynamics.ActiveReports.Chart.WallRange wallRange1 = new
DataDynamics.ActiveReports.Chart.WallRange();
DataDynamics.ActiveReports.Chart.WallRange wallRange2 = new
DataDynamics.ActiveReports.Chart.WallRange();
DataDynamics.ActiveReports.Chart.WallRange wallRange3 = new
DataDynamics.ActiveReports.Chart.WallRange();

// set the wall range properties
wallRange1.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.White);
wallRange1.Border = new DataDynamics.ActiveReports.Chart.Border(new
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, System.Drawing.Color.Black);
wallRange1.EndValue = 40;
wallRange1.PrimaryAxis =
(DataDynamics.ActiveReports.Chart.Axis)this.ChartControl1.ChartAreas[0].Axes["AxisY"];

```



```

wallRange1.StartValue = 30;
wallRange2.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Lime);
wallRange2.Border = new DataDynamics.ActiveReports.Chart.Border(new
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, System.Drawing.Color.Black);
wallRange2.EndValue = 34;
wallRange2.PrimaryAxis =
(DataDynamics.ActiveReports.Chart.Axis)this.ChartControl1.ChartAreas[0].Axes["AxisY"];
wallRange2.StartValue = 33;
wallRange3.Backdrop = new
DataDynamics.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.DarkGreen);
wallRange3.Border = new DataDynamics.ActiveReports.Chart.Border(new
DataDynamics.ActiveReports.Chart.Graphics.Line(System.Drawing.Color.Transparent, 0,
DataDynamics.ActiveReports.Chart.Graphics.LineStyle.None), 0, System.Drawing.Color.Black);
wallRange3.EndValue = 40;
wallRange3.PrimaryAxis =
(DataDynamics.ActiveReports.Chart.Axis)this.ChartControl1.ChartAreas[0].Axes["AxisZ"];
wallRange3.StartValue = 20;

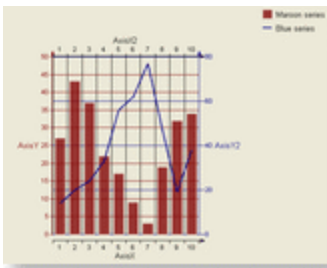
// add the wall ranges to the chart area and set wall and Z axis properties to show lines
this.chartControl1.ChartAreas[0].WallRanges.AddRange(new
DataDynamics.ActiveReports.Chart.WallRange[] {wallRange1,wallRange2,wallRange3});
this.chartControl1.ChartAreas[0].WallXY.Backdrop.Alpha = 100;
this.chartControl1.ChartAreas[0].WallXZ.Backdrop.Alpha = 100;
this.chartControl1.ChartAreas[0].WallYZ.Backdrop.Alpha = 100;
this.chartControl1.ChartAreas[0].Axes[4].MajorTick.Step = 20;
this.chartControl1.ChartAreas[0].Axes[4].Max = 60;
this.chartControl1.ChartAreas[0].Axes[4].Min = 0;
this.chartControl1.ChartAreas[0].Axes[4].Visible = true;

```

Chart Axes and Walls

Standard Axes

The Chart control provides the means to change axis settings at design time or run time. Chart axes make it possible to view and understand the data plotted in a graph.



Axis Types

Most 2D charts contain a numerical axis (AxisY) and a categorical axis (AxisX). 3D charts include another numerical axis (AxisZ). These axes are accessible at run time from the ChartArea object and allow you to control the settings for each, including scaling, labels, and various formatting properties. For any of the scaling or labeling properties you set to show up at run time, you will need to set the Visible property of the axis to True.

Changing Axis Settings

Axis settings can be changed at design time by clicking on a Chart control and using the Properties Window or at run time in code from the chart's ChartArea object.

Scaling

For normal linear scaling on a numeric axis, set the Max and Min properties for the axis, which correspond to the numerical values in the chart's data series. Also, set the Step property of the MajorTick to show the major numerical unit values. The Step property controls where labels and tick marks are shown on the numerical axis.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to set the Max, Min, and MajorTick properties.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
With Me.ChartControl1.ChartAreas(0).Axes("AxisY")
    .Max = 100
    .Min = 0
    .MajorTick.Step = 10
End With
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
this.chartControl1.ChartAreas[0].Axes["AxisY"].Max = 100;
this.chartControl1.ChartAreas[0].Axes["AxisY"].Min = 0;
this.chartControl1.ChartAreas[0].Axes["AxisY"].MajorTick.Step = 10;
```

The Chart control also supports logarithmic scaling which allows you to show the vertical spacing between two points that corresponds to the percentage of change between those numbers. You can set your numeric axis to scale logarithmically by setting the IsLogarithmic property on the axis to True and setting the Max and Min properties of the axis.

Labeling

To show labels on an axis, you will need to specify the value for the LabelsGap property, set your LabelsFont properties, and set LabelsVisible to True. These properties can be set in the AxisBase Collection editor, which is accessed at design time by clicking the ellipsis button next to the ChartAreas (Collection) property, then the Axes (Collection) property of the ChartArea.

Tip: Labels render first, and then the chart fills in the remaining area, so be sure to make the chart large enough if you use angled labels.

You can specify strings to be used for the labels instead of numerical values on an axis by using the Labels collection property at design time or assigning a string array to the Labels property at run time. You can also specify whether you want your axis labels to appear on the outside or inside of the axis line using the LabelsInside property. By default, labels appear outside the axis line.

Secondary Axes

By default, a Chart object includes secondary X and Y axes (AxisX2 and AxisY2). At design time or run time, you can specify a secondary axis to plot data against by setting all of the appropriate properties for AxisX2 or AxisY2, including the Visible property.

If you want to use two axes to show the same data as it appears on two different scales, you can set the primary axis to show the actual data value scale, for example, and set the secondary axis to show a logarithmic scale.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.

2. Add code to the handler to create a primary axis with actual data, and a secondary axis with a logarithmic scale.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' set properties for AxisY (primary axis)
With Me.ChartControl1.ChartAreas(0).Axes("AxisY")
    .Max = 25
    .Min = 0
    .MajorTick.Step = 5
End With

' set properties for AxisY2 (secondary Y axis)
With Me.ChartControl1.ChartAreas(0).Axes("AxisY2")
    .Max = 1000
    .Min = 0
    .MajorTick.Step = 200
' set the scaling for the secondary axis to logarithmic
    .AxisType = DataDynamics.ActiveReports.Chart.AxisType.Logarithmic.Visible = True
End With
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

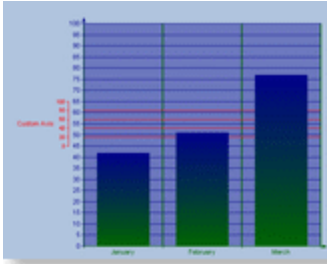
```
// set properties for AxisY (primary axis)
this.chartControl1.ChartAreas[0].Axes["AxisY"].Max = 25;
this.chartControl1.ChartAreas[0].Axes["AxisY"].Min = 0;
this.chartControl1.ChartAreas[0].Axes["AxisY"].MajorTick.Step = 5;

// set properties for AxisY2 (secondary Y axis)
this.chartControl1.ChartAreas[0].Axes["AxisY2"].Max = 1000;
this.chartControl1.ChartAreas[0].Axes["AxisY2"].Min = 0;
this.chartControl1.ChartAreas[0].Axes["AxisY2"].MajorTick.Step = 200;
// set the axis type for the secondary axis to logarithmic
this.chartControl1.ChartAreas[0].Axes["AxisY2"].AxisType =
DataDynamics.ActiveReports.Chart.AxisType.Logarithmic;
this.chartControl1.ChartAreas[0].Axes["AxisY2"].Visible = true;
```

Custom Axes

The Chart control supports the creation of additional custom axes through the use of the chart's CustomAxes collection. Once a custom axis has been added to the collection, in addition to setting the normal axis properties, you will need to set the following properties:

- Parent - The Parent property allows you to choose the primary or secondary axis on which your custom axis resides.
- PlacementLength - The PlacementLength property allows you to set the length of the custom axis in proportion to the Min and Max property values you have already set for the parent axis.
- PlacementLocation - The PlacementLocation property allows you to set the starting location value for the custom axis to appear in relation to the parent axis.



The following code sample demonstrates creating a custom axis, adding it to the Axes collection for the ChartArea, and setting its properties.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a custom axis, add it to the chart area, and set its properties.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the custom axis and add it to the ChartArea's Axes collection
Dim customAxisY As New DataDynamics.ActiveReports.Chart.CustomAxis
Me.ChartControl1.ChartAreas(0).Axes.Add(customAxisY)

' set the basic axis properties for customAxisY
customAxisY.LabelFont = New DataDynamics.ActiveReports.Chart.FontInfo(Color.Red, New
Font("Arial", 7.0!))
customAxisY.LabelsGap = 1
customAxisY.LabelsVisible = True
customAxisY.Line = New DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red)
customAxisY.MajorTick = New DataDynamics.ActiveReports.Chart.Tick(New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1), New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1), 1, 2.0F, True)
customAxisY.MajorTick.GridLine = New
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1, LineStyle.Solid)
customAxisY.MajorTick.Visible = True
customAxisY.Max = 5
customAxisY.MaxDerived = False
customAxisY.Min = 0
customAxisY.Visible = True

' set the special custom axis properties
customAxisY.Parent = ((CType(Me.ChartControl1.ChartAreas(0).Axes("AxisY"),
DataDynamics.ActiveReports.Chart.Axis))
customAxisY.PlacementLength = 20
customAxisY.PlacementLocation = 30
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the custom axis and add it to the ChartArea's Axes collection
DataDynamics.ActiveReports.Chart.CustomAxis customAxisY = new
DataDynamics.ActiveReports.Chart.CustomAxis();
this.chartControl1.ChartAreas[0].Axes.Add(customAxisY);

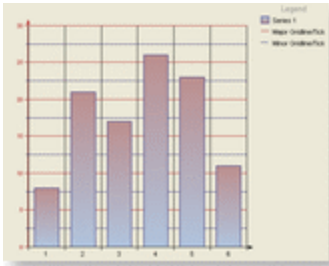
// set the basic axis properties for customAxisY
customAxisY.LabelFont = new DataDynamics.ActiveReports.Chart.FontInfo(Color.Red, new
```

```
Font("Arial", 7F, FontStyle.Regular, GraphicsUnit.Point, ((System.Byte)(0)));
customAxisY.LabelsGap = 1;
customAxisY.LabelsVisible = true;
customAxisY.Line = new DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red);
customAxisY.MajorTick = new DataDynamics.ActiveReports.Chart.Tick(new
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1), new
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1), 1, 2F, true);
customAxisY.MajorTick.GridLine = new
DataDynamics.ActiveReports.Chart.Graphics.Line(Color.Red, 1, LineStyle.Solid);
customAxisY.MajorTick.Visible = true;
customAxisY.Max = 5;
customAxisY.MaxDerived = false;
customAxisY.Min = 0;
customAxisY.Visible = true;

// set the special custom axis properties
customAxisY.Parent =
(DataDynamics.ActiveReports.Chart.Axis)this.ChartControl1.ChartAreas[0].Axes["AxisY"];
customAxisY.PlacementLength = 20;
customAxisY.PlacementLocation = 30;
```

Gridlines and Tick Marks

Gridlines and tick marks are generally used to help increase the readability of a chart.



Types

There are two kinds of gridlines and tick marks in the Chart control: major and minor. The properties for the major gridlines and tick marks are set on the MajorTick object of the particular axis and the properties for minor gridlines and ticks are set on the MinorTick object of the axis. The location for any labels shown for the axis are determined by the Step property of the MajorTick object.

Step and TickLength

For either the MajorTick or MinorTick objects, you can define where the tick marks and gridlines will appear by setting the Step property. The TickLength property allows you to set how far outside of the axis line the tick mark will extend.

Visible

To make any defined major or minor tick marks to show up at design time or run time, the Visible property of the MajorTick or MinorTick object must be set to True. To show major or minor gridlines at design time or run time, the Visible property of the WallXY object as well as that of the MajorTick or MinorTick object must be set to True.

Chart Data

Bound Data

The Chart control has its own data source which is distinct from the report data source. To access the chart's data source, click the Data Source verb which appears below the Properties window when the chart is selected on the report. If the Data Source verb does not appear, see **Access the Chart Wizard and Data Source** for help.

Unbound Data

The Chart control allows you to set the data source for a chart control, series, or data points collection at run time.

Any of the following objects can be used as data sources:

- Dataset
- Dataset Column
- Data Table
- SqlCommand/OleDbCommand
- SqlDataAdapter/OleDbDataAdapter
- Array
- XML data

Below are some examples of binding to different data sources at run time.

Dataset

The Chart control's DataSource property can be set to a dataset at run time. The following code demonstrates how to set up a dataset, set the DataSource property to the dataset, create a series, and set the ValueMembersY property to the dataset expression at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a data source and bind a series to a dataset expression.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the series
Dim s As New DataDynamics.ActiveReports.Chart.Series
Dim m_cnnString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb; Persist Security Info=False"
Dim m_cnn As New System.Data.OleDb.OleDbConnection(m_cnnString)
Dim oDBAdapter As System.Data.OleDb.OleDbDataAdapter

' create the dataset
Dim oDS As DataSet
oDBAdapter = New System.Data.OleDb.OleDbDataAdapter("SELECT ShipCountry, SUM(Freight)
AS Expr1 FROM Orders GROUP BY ShipCountry", m_cnnString)
oDS = New DataSet
oDBAdapter.Fill(oDS, "Expr1")

' set the DataSource and ValueMembersY properties
Me.ChartControl1.DataSource = oDS
s.ValueMembersY = "Expr1"

Me.ChartControl1.Series.Add(s)
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the series
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb;Persist Security Info=False";
System.Data.OleDb.OleDbConnection m_cnn = new
System.Data.OleDb.OleDbConnection(m_cnnString);
System.Data.OleDb.OleDbDataAdapter oDBAdapter;

// create the dataset
System.Data.DataSet oDS;
oDBAdapter = new System.Data.OleDb.OleDbDataAdapter("SELECT ShipCountry, SUM(Freight)
AS Expr1 FROM Orders GROUP BY ShipCountry", m_cnnString);
oDS = new System.Data.DataSet();
oDBAdapter.Fill(oDS, "Expr1");

// set the DataSource and ValueMembersY properties
this.chartControl1.DataSource = oDS;
s.ValueMembersY = "Expr1";

this.chartControl1.Series.Add(s);
```

Dataset Column

In the Chart control, the ValueMembersX and ValueMembersY properties of a series can be set to a dataset column. The following code demonstrates how to create a series, set up a dataset, set the DataSource property to the dataset, and set the ValueMembersY and ValueMembersX properties to dataset columns at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a data source and bind a series to dataset columns.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the series
Dim s As New DataDynamics.ActiveReports.Chart.Series
Dim m_cnnString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb; Persist Security Info=False"
Dim m_cnn As New System.Data.OleDb.OleDbConnection(m_cnnString)
Dim oDBAdapter As System.Data.OleDb.OleDbDataAdapter

' create the dataset
Dim oDS As DataSet
oDBAdapter = New System.Data.OleDb.OleDbDataAdapter("SELECT * from Orders WHERE
OrderDate < #08/17/1994#", m_cnnString)
oDS = New DataSet
oDBAdapter.Fill(oDS, "Orders")

' set the DataSource, ValueMembersY, and ValueMembersX properties
Me.ChartControl1.DataSource = oDS
Me.ChartControl1.Series.Add(s)
Me.ChartControl1.Series(0).ValueMembersY = oDS.Tables("Orders").Columns(7).ColumnName
Me.ChartControl1.Series(0).ValueMemberX = oDS.Tables("Orders").Columns(8).ColumnName
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the series
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb;Persist Security Info=False";
System.Data.OleDb.OleDbConnection m_cnn = new
System.Data.OleDb.OleDbConnection(m_cnnString);
System.Data.OleDb.OleDbDataAdapter oDBAdapter;

// create the dataset
System.Data.DataSet oDS;
oDBAdapter = new System.Data.OleDb.OleDbDataAdapter("SELECT * from Orders WHERE
OrderDate < #08/17/1994#", m_cnnString);
oDS = new System.Data.DataSet();
oDBAdapter.Fill(oDS, "Orders");

// set the DataSource, ValueMembersY, and ValueMembersX properties
this.chartControll1.DataSource = oDS;
this.chartControll1.Series.Add(s);
this.chartControll1.Series[0].ValueMembersY =
oDS.Tables["Orders"].Columns[7].ColumnName;
this.chartControll1.Series[0].ValueMemberX = oDS.Tables["Orders"].Columns[8].ColumnName;
```

Data Command

You can set a chart's data source to a SqlCommand or OleDbCommand. The following code demonstrates how to create a series, create an OleDbCommand, set the DataSource property to the data command, and set the ValueMembersY property for the series at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a data source and bind a series.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

```
' create the series
Dim s As New DataDynamics.ActiveReports.Chart.Series
Dim m_cnnString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb;Persist Security Info=False"
Dim m_cnn As New System.Data.OleDb.OleDbConnection(m_cnnString)
Dim query As String = "SELECT ShipCountry, SUM(Freight) AS Expr1 FROM Orders GROUP BY
ShipCountry"

' create the OleDbCommand and open the connection
Dim command As New System.Data.OleDb.OleDbCommand(query, m_cnn)
command.Connection.Open()

' set the DataSource and ValueMembersY properties
Me.ChartControll1.DataSource = command
Me.ChartControll1.Series.Add(s)
Me.ChartControll1.Series(0).ValueMembersY = "Expr1"

' close the connection
m_cnn.Close()
```


To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
// create the series
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
string m_cnnString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:/Northwind.mdb;Persist Security Info=False";
System.Data.OleDb.OleDbConnection m_cnn = new
System.Data.OleDb.OleDbConnection(m_cnnString);
string query = "SELECT ShipCountry, SUM(Freight) AS Expr1 FROM Orders GROUP BY
ShipCountry";

// create the OleDbCommand and open the connection
System.Data.OleDb.OleDbCommand command = new System.Data.OleDb.OleDbCommand(query,
m_cnn);
command.Connection.Open();

// set the DataSource and ValueMembersY properties
this.chartControl1.DataSource = command;
this.chartControl1.Series.Add(s);
this.chartControl1.Series[0].ValueMembersY = "Expr1";

// close the connection
m_cnn.Close();
```

Array

The Chart control allows you to set the data source for the data points collection to an array. The following code demonstrates how to create a series, create an array, and use the DataBindY method to set the data source for the data points collection at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a series, an array, and a data source.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the section Format event.**

```
' create the series
Dim s As New DataDynamics.ActiveReports.Chart.Series

' create the array
Dim a As Double() = {1, 4, 2, 6, 3, 3, 4, 7}

' set the data source for the data points collection
Me.ChartControl1.Series.Add(s)
Me.ChartControl1.Series(0).Points.DataBindY(a)
```

To write the code in C#**C# code. Paste INSIDE the section Format event.**

```
// create the series
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();

// create the array
```

```
double [] a = {1,4,2,6,3,3,4,7};

// set the data source for the data points collection
this.chartControll1.Series.Add(s);
this.chartControll1.Series[0].Points.DataBindY(a);
```

Calculated Series

You can easily create a calculated series based on the values of one or more series by setting the ValueMembersY property of a series to a formula. To reference a series in the formula, use the name of the series. The following code demonstrates how to create two series, one bound to a data array and the other bound to a formula based on the Y values of the first series.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to create a data bound series and a calculated series.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the FetchData event.

```
Dim s As New DataDynamics.ActiveReports.Chart.Series
Dim cS As New DataDynamics.ActiveReports.Chart.Series
Dim a As Double() = {1, 4, 2, 6, 3, 3, 4, 7}

Me.ChartControll1.Series.AddRange(New DataDynamics.ActiveReports.Chart.Series() {s, cS})
Me.ChartControll1.Series(0).Points.DataBindY(a)
Me.ChartControll1.Series(0).Name = "Series1"
Me.ChartControll1.Series(1).ValueMembersY = "Series1.Y[0]+10"
```

To write the code in C#

C# code. Paste INSIDE the FetchData event.

```
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
DataDynamics.ActiveReports.Chart.Series cS = new
DataDynamics.ActiveReports.Chart.Series();
double [] a = { 1,4,2,6,3,3,4,7};

this.chartControll1.Series.AddRange(new DataDynamics.ActiveReports.Chart.Series[] {s,
cS});
this.chartControll1.Series[0].Name = "Series1";
this.chartControll1.Series[0].Points.DataBindY(a);
this.chartControll1.Series[1].ValueMembersY = "Series1.Y[0]+10";
```

Sequence Series

Set a sequence series by specifying the minimum value, maximum value, and step for the series. The following code shows how to set the ValueMembersY property at run time to create a sequence series.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to add a series to the chart and set its members.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
Dim s As New DataDynamics.ActiveReports.Chart.Series
```

```
Me.ChartControl1.Series.Add(s)
Me.ChartControl1.Series(0).ValueMembersY = "sequence(12,48,4)"
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
this.chartControl1.Series.Add(s);
this.chartControl1.Series[0].ValueMembersY = "sequence(12,48,4)";
```

XML Data

The Chart control allows you to set the data source to an XML document. The following code demonstrates how to create a series and to set the ValueMembersY and ValueMemberX properties at run time.

1. In design view of the report, double-click the section where you placed your chart. This creates a Format event handling method for the section.
2. Add code to the handler to add a series to the chart and set its members.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the section Format event.

```
' create the series
Dim s As New DataDynamics.ActiveReports.Chart.Series()
Dim xmlDS As New
DataDynamics.ActiveReports.DataSources.XMLDataSource("c:/customer.xml", "//CUSTOMER")

' set the DataSource, ValueMembersY and ValueMemberX properties
Me.ChartControl1.DataSource = xmlDS
Me.ChartControl1.Series.Add(s)
Me.ChartControl1.Series(0).ValueMembersY = "xpath:ORDER/ITEM/PRICE"
Me.ChartControl1.Series(0).ValueMemberX = "xpath:ORDER/ITEM/AUTHOR"
```

To write the code in C#

C# code. Paste INSIDE the section Format event.

```
// create the series
DataDynamics.ActiveReports.Chart.Series s = new
DataDynamics.ActiveReports.Chart.Series();
DataDynamics.ActiveReports.DataSources.XMLDataSource xmlDS = new
XMLDataSource("c:/customer.xml", "//CUSTOMER");


// set the DataSource, ValueMembersY and ValueMemberX properties
this.chartControl1.DataSource = xmlDS;
this.chartControl1.Series.Add(s);
this.chartControl1.Series[0].ValueMembersY = "xpath:ORDER/ITEM/PRICE";
this.chartControl1.Series[0].ValueMemberX = "xpath:ORDER/ITEM/AUTHOR";
```

RichText


The RichText control gives you an enormous amount of control over the appearance of text in the report. Unlike the TextBox control which applies formatting to the entire contents of the control, the RichText control allows you to apply selective formatting to various areas of text within the control.

For example, the **Find** method allows you to find specific words or characters in the control, while the various selection properties allow you to select text and change its formatting in a dozen ways.

You can set the control's text directly using the **Text**, **Html**, or **RTF** properties, or you can load it from a plain text, RTF or HTML file or stream using the **Load** method.

 **Tip:** In order to show special characters in an html file loaded into the control, use the character entity reference (for example, **è** for è or **&** for &).

Use the **InsertField** and **ReplaceField** methods for field merging reports, such as the mail merge report demonstrated in the **Mail Merge with RichText** walkthrough. The RichTextBox now automatically binds inserted fields to the report's fields collection, so you only need to use InsertField and ReplaceField for special cases such as conditional values or system dates.

 **Note:** If you have trouble loading a file at design time, be sure that you are not in edit mode. You are in edit mode if your cursor appears inside the control.

The following is a list of all of the HTML tags that can be used with the RichText control. Unsupported tags are ignored. Please note that W3C conventions are strictly observed.

Supported HTML Tags

| Tag | Description | Attributes |
|----------------|---------------------------------|---|
| | Bold | none |
| <I> | Italic | none |
| <P> | Paragraph | align, style |
| | Strong (looks like bold) | none |
| <BIG> | Big | none |
| <SMALL> | Small | none |
| <PRE> | Preformatted | none |
| | Font | face, size, color, style (see notes for style attributes) |
| <BODY> | The body tag | background, text, leftmargin |
| <H1> - <H6> | Heading levels one through six | none |
| | Line break | none |
| | Emphasized (looks like Italics) | none |
| <U> | Underlined | none |
| | Image | align, height, src, width |
| <SUP> | Superscript | none |
| <SUB> | Subscript | none |
| <CENTER> | Center alignment | none |
| <TABLE> | Table | align, border, cellpadding, cellspacing, height, style, width |
| <TR> | Table row | align |
| <TH> | Table head | none |
| <TD> | Table datum | align, border, colspan, rowspan, width |

| | | |
|----------|----------------|---|
| | List item | none (nested levels automatically use disc, then circle, then square bullets) |
| | Ordered list | type |
| | Unordered list | type |
| <STRIKE> | Strike through | none |

The **style** attribute of , <P>, and <TABLE> supports the following properties:

Supported Style Attribute Properties

- border-bottom
- border-bottom-width
- border-color
- border-left
- border-left-width
- border-right
- border-right-width
- border-style
- border-top
- border-top-width
- border-width
- font-family
- font-size
- height
- line-height
- margin-bottom
- margin-left
- margin-right
- margin-top
- padding-bottom
- padding-left
- padding-right
- padding-top
- table-layout
- text-align
- text-indent
- width

Grouping Data

When you add a pair of group header and group footer sections to a report, the new sections appear immediately above and below the detail section.



Note: You cannot add a header section without a corresponding footer section. If you try to do so in code, the results are highly unstable.

You can, however, hide one of the sections. To hide a section, set its **Visible** property to **False**, or set its **Height** property to **0**.

Set group header section's **DataField** property to the field on which you want to group the report. For best results, in the SQL query, order the data by the grouping field.

When you run the report, it renders the group header, followed by all related instances of the detail section, and then the group footer. It renders a new group header section for each instance of the grouping field.

Controls in the group header render once for each instance of the group, so this is a good place for column header labels to describe the data in the detail fields.

Group options

With the group header selected in the Properties window, there are a number of properties that allow you to control group header behavior.

For more information on columnar reporting, see **Columnar Reports**. For more information on KeepTogether options, see **KeepTogether Options**.

| Property | Description |
|-------------------------|---|
| ColumnGroupKeepTogether | If possible, keeps grouped items together in a single column. |
| ColumnLayout | If True, lays out the group header with the same number of columns as the detail section. |

| | |
|-------------------|---|
| DataField | Sets the field on which to group the report. |
| GroupKeepTogether | If possible, keeps grouped items together on a single page. |
| KeepTogether | If possible, keeps the section together on a single page. |
| NewColumn | Tells the group to begin a new column before or after the group field changes. |
| NewPage | Tells the group to begin a new page before or after the group field changes. |
| RepeatStyle | Allows you to have the group header render on each page or column for long groups. |
| UnderlayNext | If True, renders the group header as a layer underneath the following section. This is useful if you want to render a group header control in the same row as data in a nested group header or in the detail section. Take care to leave the BackColor property set to Transparent in the following section, or the group header is hidden. |

Multiple Groupings

You can nest group header and footer pairs and group each on a different field.

Important: As with any group header and footer pair, group your data on the fields that you specify in the **DataField** property of the group header, but in the order of your groups. For example:

```
SELECT * FROM Customers ORDER BY GroupHeader1DataField, GroupHeader2DataField,
GroupHeader3DataField
```

See the image below for the order in which report sections appear on the report. GroupHeader1 in the image was added first and appears above the other two group headers, while GroupFooter1, its partner, appears below the other two group footers.



When you run a report with multiple groupings like the one above, the sections print in an order like the following:

1. **ReportHeader1** prints once and does not repeat.
2. **PageHeader1** prints once at the top of each page.
3. **GroupHeader1** prints once for the first value its DataField returns.
4. **GroupHeader2** prints once for the first value its DataField returns within the context of GroupHeader1's DataField value.
5. **GroupHeader3** prints once for the first value its DataField returns within the context of GroupHeader2's DataField value.
6. **Detail1** prints once for each record that falls within the context of GroupHeader3's DataField value.
7. **GroupFooter3** prints once at the end of the records that fall within the context of GroupHeader3's DataField value.
8. **GroupHeader3** may print again, if more values return within the context of GroupHeader2's DataField value.
9. Each time GroupHeader3 prints again, it is followed by Detail1 (once for each related record) and GroupFooter3.

10. **GroupFooter2** prints once after GroupFooter3.
11. **GroupHeader2** may print again, if more values return within the context of GroupHeader1's DataField value.
12. Each time GroupHeader2 prints again, it is followed by Detail1 (once for each related record) and GroupFooter2.
13. **GroupFooter1** prints once after GroupFooter2.
14. **GroupHeader1** prints once for the second value its DataField returns, followed by GroupHeader2, and so on in a pattern similar to the one above.
15. **PageFooter1** prints once at the bottom of each page. (Its position within groupings varies.)
16. **ReportFooter1** prints once at the end of the report.

You can add up to 32 groupings in one report. With many groupings, you might find the need to rearrange the order of your groups. If your report has more than one group and you right-click the report surface, you will see an extra option in the context menu: **Reorder Groups**. Select that option to open the Group Order window, in which you can drag and drop groups to put them in any order you want.



Subreports

In ActiveReports, you can embed a report in another report using the Subreport control. Once you place the Subreport control on a report, you attach a report object to it in code. You can pass parameters to the subreport from the main report, ensuring that data related to the main report displays in each instance of the subreport.

Since they render inside the main report, subreports are disconnected from any concept of a printed page. For this reason, page-dependent features are not supported for use in subreports. Keep any such logic in the main report. Page-related concepts that are not supported in subreports include:

- Page numbers
- Page header and footer sections (delete these sections to save processing time)
- KeepTogether properties
- GroupKeepTogether properties
- NewPage properties

Because of the high overhead of running a second report and embedding it in the first, it is generally best to instead use **grouping** wherever possible. If grouping cannot accommodate your particular report, use subreports. Some uses of subreports include:

- Repeating groups
- Relational data
- Multiple data sources
- Multiple detail sections

As a best practice, create an instance of the report for your Subreport control in the **ReportStart** event of the main report, and then dispose of it in the **ReportEnd** event. In this way, you are creating only one subreport instance when you run the main report.

If you instantiate the subreport in a section Format event, it creates a new instance of the subreport each time the section processes. This consumes a lot of memory and processing time, especially in a report that processes a large amount of data.

Report Events

Events that are Raised Only Once

The following events are all of the events that are raised only once during a report's processing. These events are raised at the beginning and at the end of the report processing cycle.

ReportStart

This event is raised before the `DataInitialize` event. Use this event to initialize any objects or variables needed while running a report. Also use this event to set any Subreport control objects to a new instance of the report assigned to the Subreport control. Do not add items dynamically to a report once this event has finished.

DataInitialize

This event is raised after `ReportStart`. Use it to add custom fields to the report's `Fields` collection. Custom fields can be added to a bound report (one that uses a `DataControl` to connect and retrieve records) or an unbound report (one that does not depend on a data control to get its records). In a bound report the dataset is opened and the dataset fields are added to the custom fields collection, then the `DataInitialize` event is raised so new custom fields can be added. The `DataInitialize` event can also be used to make adjustments to the `DataSource` or to set up database connectivity.

NoData

This event is raised if the report's data source does not return any records and if there is no data to be processed. Use this event either to cancel the report or to print a page with a message informing the user that the report did not return any records to print.

ReportEnd

This event is raised after the report finishes processing. Use this event to close or free any objects that you were using while running a report in unbound mode, or to display information or messages to the end user. This event can also be used to export reports.

Events that are Raised Multiple Times

The following ActiveReports events are raised multiple times during a report's processing.

FetchData

This event is raised every time a new record is processed. The `FetchData` has an `EOF` parameter indicating whether the `FetchData` event should be raised. This parameter is not the same as the `Recordset`'s `EOF` property and is defaulted to `True`. When working with bound reports (reports using a `DataControl`), the `EOF` parameter is automatically set by the report; however, when working with unbound reports this parameter needs to be controlled manually.

Use the `FetchData` event with unbound reports to set the values of custom fields that were added in the `DataInitialize` event or with bound reports to perform special functions, such as combining fields together or performing calculations. The `FetchData` event should not have any references to controls on the report.

If you need to use a value from a `Dataset` with a control in the `Detail` section, set a variable in the `FetchData` event and use the variable in the section's `Format` event to set the value for the control. Please note that this method of setting a variable in the `FetchData` event and using it to set a control's value is only supported in the `Detail_Format` event.

Also use the `FetchData` event to increment counters when working with arrays or collections.

PageStart


This event fires before a page is rendered. Use this event to initialize any variables needed for each page when running an unbound report.

PageEnd

This event is raised after each page in the report is rendered. Use this event to update any variables needed for each page when running an unbound report.

When Bound and Unbound Data Values are Set


1. The Fields collection is populated from the dataset that is bound to the report after the DataInitialize event is raised. (In an unbound report, the Fields collection values are not set to anything at this point.)
2. The FetchData event is raised, giving the user a chance to modify the Fields collection.
3. Any fields that are bound have the values transferred over.
4. The Format event is raised.

 **Note:** See **Scripting** to learn about the difference in script and code-behind event handler method definition.

Section 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-raising sequence is dynamic in order to accommodate individual report demands. The only guaranteed sequence is that a section's Format event is raised before the BeforePrint event, which in turn occurs before the AfterPrint event but not necessarily all together. Reports should not be designed to rely on these events being raised in immediate succession.

 **Important:** Never reference the report's Fields collection in these section events. Only reference the Fields collection in the **DataInitialize** and **FetchData** events.

Format event

ActiveReports raises this event 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 in which you can change the section's height. Use this section to set or change the properties of any controls or the section itself.

Also use the Format event to pass information, such as an SQL String, to a Subreport.

If the CanGrow or CanShrink property is True for the section or any control within the section, all of the growing and shrinking takes place in the Format event. Because of this, you cannot obtain information about a control or section's height in this event.

Because a section's height is unknown until the Format event finishes, it is possible for a section's Format event to be raised while the report is on a page to which the section is not rendered. For example, the Detail Format event is raised but the section is too large to fit on the page. This causes the PageFooter events and the PageEnd event to be raised on the current page, and the PageStart, any other Header events, and possibly the FetchData event to be raised before the section is rendered to the canvas on the next page.


BeforePrint event

ActiveReports raises this event before the section is rendered to the page.

The growing and shrinking of the section and its controls have already taken place. Therefore, you can use this event to get an accurate height of the section and its controls. You can modify values and resize controls in the BeforePrint event, but you cannot modify the height of the section itself.

Also use this event to do page-specific formatting since the report knows which page the section will be rendered to

when this event is raised. Once this event has finished, the section cannot be changed in any way because the section is rendered to the canvas immediately after this event.

 **Note:** If a section contains the SubReport control that occupies more than one page, the SubReport gets split into smaller parts at rendering. In this case, you can use the BeforePrint event - it will fire multiple times to get the height of each part of the rendered SubReport.

AfterPrint event

ActiveReports raises this event 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. This event is still useful, however, if you want to draw on the page after text has already been rendered to it.

Sequence of Events

Intelligent, multi-threaded, single-pass processing enables ActiveReports to surpass other reports in processing and output generation speed. ActiveReports processes and renders each page as soon as the page is ready. If a page has unknown data elements or its layout is not final, it places the page in cache until the data is available.

Summary fields and KeepTogether constraints are two reasons why a page might not render immediately. 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 are delayed until all of the data is read.

There are eleven report events in the code behind an ActiveReport, or seven in an ActiveReport script.

|  (Declarations) | |
|--|--|
|  DataInitialize | |
|  DataSourceChanged | |
|  FetchData | |
|  NoData | |
|  PageEnd | |
|  PageStart |  ReportStart |
|  ParameterUIClosed |  ReportEnd |
|  PrintAborted |  PageStart |
|  PrintProgress |  PageEnd |
|  ReportEnd |  DataInitialize |
|  ReportStart |  FetchData |
| |  NoData |

Because there are so many ways in which you can customize your reports, not all reports execute in the same way. However, when you run a report, this is generally what happens:

1. ActiveReports raises the **ReportStart** event. The report validates any changes made to the report structure in ReportStart. In some cases, data source properties raise the **DataInitialize** event.
2. Printer settings are applied. If none are specified, the local machine's default printer settings are used.
3. If the **DataInitialize** event was not already raised, ActiveReports raises it and opens the data source.
4. If the data source contains **parameters** with unset values and the **ShowParameterUI** property is set to **True**, ActiveReports displays a parameters dialog to request values from the user.
5. Closing the dialog raises the **ParameterUIClosed** event. If the report is a subreport that requires parameters, ActiveReports binds the subreport parameters to any fields in the parent report.
6. ActiveReports raises the **FetchData** event.
7. If there is no data, the **NoData** event is raised.
8. The **PageStart** event raises, and then raises again after each **PageEnd** event until the final page.
9. Group sections are bound and sections begin rendering on pages.

10. ActiveReports raises **Section Events** to process sections in (roughly) the following order:
 - Report header
 - Page header
 - Group header
 - Detail
 - Group footer
 - Page footer
 - Report footer
11. After each event, ActiveReports checks the **Cancel** flag to ensure that it should continue.
12. Other events may raise, depending on the report logic.
13. The **PageEnd** event raises after each page becomes full, and the **PageStart** raises if the report has not finished.
14. Finally, ActiveReports raises the **ReportEnd** event.

Unbound Reporting

ActiveReports allows you to bind reports to any type of data source, including arrays. You can create a report without setting its data source, then load the data into the control at run time. Use the ReportStart event to set up your data source, the ReportEnd event to close it, the DataInitialize event to create your fields collection, and the FetchData event to populate it.

Use the ReportStart event to connect the report to a data source

1. Double-click in the gray area below the report to create an event-handling method for the ReportStart event.
2. Add code to the handler to:
 - Set the database path (place this code above the ReportStart event)
 - Set the data source connection string
 - Set the data source SQL query
 - Open the connection to create the DataReader

The following examples show what the code for the method looks like.

Add using or Imports statements for System.Data and System.Data.OleDb.

To create a GetDatabasePath method in Visual Basic.NET

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Private Function getDatabasePath() As String
    Dim regKey As Microsoft.Win32.RegistryKey
    regKey = Microsoft.Win32.Registry.LocalMachine
    regKey = regKey.CreateSubKey _
        ("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB")
    getDatabasePath = CType(regKey.GetValue(""), String)
End Function
```

```
Private conn As OleDbConnection
Private reader As OleDbDataReader
Private cmd As OleDbCommand
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Private Sub rptYourReportName_ReportStart(ByVal sender As Object, ByVal e As
System.EventArgs) _
    Handles MyBase.ReportStart
    Dim dbPath As String = getDatabasePath()
```

```

Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + dbPath
+ "\\NWIND.mdb"
conn = New OleDbConnection(connString)
cmd = New OleDbCommand("SELECT * FROM categories INNER JOIN products ON _
categories.categoryid = products.categoryid ORDER BY products.categoryid,
products.productid", conn)
conn.Open()
reader = cmd.ExecuteReader()
End Sub

```

To create a `GetDatabasePath` method in C#

C# code. Paste **JUST ABOVE** the `ReportStart` event.

```

private string getDatabasePath()
{
    Microsoft.Win32.RegistryKey regKey = Microsoft.Win32.Registry.LocalMachine;
    regKey = regKey.CreateSubKey("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB");
    return ((string) (regKey.GetValue("")));
}

private static OleDbConnection conn;
private static OleDbDataReader reader;

```

C# code. Paste **INSIDE** the `ReportStart` event.

```

string dbPath = getDatabasePath();
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + dbPath +
"\\NWIND.mdb";
conn = new OleDbConnection(connString);
OleDbCommand cmd = new OleDbCommand("SELECT * FROM categories INNER JOIN products ON
categories.categoryid = products.categoryid ORDER BY products.categoryid,
products.productid", conn);
conn.Open();
reader = cmd.ExecuteReader();

```

Use the `ReportEnd` event to close the data connection

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET

1. Right-click in any section of the design window of the report and select **View Code**.
2. At the top left of the code view, click the drop-down arrow and select (***rptYourReportName* Events**).
3. At the top right of the code window, click the drop-down arrow and select **ReportEnd**. This creates an event-handling method for the report's `ReportEnd` event.
4. Add code to the handler to close the data connection

Visual Basic.NET code. Paste **INSIDE** the `ReportEnd` event.

```

reader.Close()
conn.Close()

```

To write the code in C#

1. Click in the gray area below the report to select it.
2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **ReportEnd**. This creates an event-handling method for the report's `ReportEnd` event.

4. Add code to the handler to close the data connection

C# code. Paste INSIDE the ReportEnd event.

```
reader.Close();  
conn.Close();
```

Use the DataInitialize event to add fields

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET

1. Right-click in any section of the design view of the report and select **View Code**.
2. At the top left of the code view, click the drop-down arrow and select (**rptYourReportName Events**).
3. At the top right of the code window, click the drop-down arrow and select **DataInitialize**. This creates an event-handling method for the report's **DataInitialize** event.
4. Add code to the handler to add fields to the report's fields collection.

Visual Basic.NET code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryName")  
Fields.Add("ProductName")  
Fields.Add("UnitsInStock")  
Fields.Add("Description")
```

To write the code in C#

1. Click in the gray area below the report to select it.
2. In the Properties window, click the events icon to display available events for the report.
3. Double-click **DataInitialize**. This creates an event-handling method for the report's **DataInitialize** event.
4. Add code to the handler to add fields to the report's fields collection.

C# code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryName");  
Fields.Add("ProductName");  
Fields.Add("UnitsInStock");  
Fields.Add("Description");
```

Use the FetchData event to populate fields

Reference the Fields collection only in the **DataInitialize** and **FetchData** events.

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET

1. Right-click in any section of the design window of the report, and select **View Code** to display the code view for the report.
2. At the top left of the code view, click the drop-down arrow and select (**rptYourReportName Events**).
3. At the top right of the code window, click the drop-down arrow and select **FetchData**. This creates an event-handling method for the report's **FetchData** event.
4. Add code to the handler to retrieve information to populate the report fields.

Visual Basic.NET code. Paste INSIDE the FetchData event.

```
Try  
    reader.Read()
```

```
Me.Fields("CategoryName").Value = reader("CategoryName")
Me.Fields("ProductName").Value = reader("ProductName")
Me.Fields("UnitsInStock").Value = reader("UnitsInStock")
Me.Fields("Description").Value = reader("Description")
eArgs.EOF = False
Catch ex As Exception
    eArgs.EOF = True
End Try
```

To write the code in C#

1. Click in the gray area below the report to select it.
2. In the Properties window, click the events icon to display available events for the report.
3. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
4. Add code to the handler to retrieve information to populate the report fields.

C# code. Paste **INSIDE** the FetchData event.

```
try
{
    reader.Read();
    Fields["CategoryName"].Value = reader["CategoryName"].ToString();
    Fields["ProductName"].Value = reader["ProductName"].ToString();
    Fields["UnitsInStock"].Value = reader["UnitsInStock"].ToString();
    Fields["Description"].Value = reader["Description"].ToString();
    eArgs.EOF = false;
}
catch
{
    eArgs.EOF = true;
}
```

Optimizing ActiveReports

Optimization can be crucial for large reports (i.e. over 100 pages). Here is some information which will help you to achieve the best possible results for such reports. To optimize ActiveReports for the web, please refer to the memory considerations section.

Memory Considerations

- **Images** Limit the use of large images when exporting to RTF and TIFF formats. Note that even one image uses a lot of memory if it's repeated on every page of a very long report exported to TIFF or RTF. If you are not exporting, or if you are exporting to Excel, PDF, or HTML, repeated images are stored only once to save memory, but the comparison necessary to detect duplicate images slows the processing time for the report.
- **Subreports** Limit the use of subreports in repeating sections because each subreport instance consumes memory. For example, consider that a subreport in the Detail section of a report in which the Detail section is repeated 2,000 times will have 2,000 instances of the subreport. Nested subreports will compound the number of instances. If you need to use subreports in repeating sections, instantiate them in the ReportStart event instead of the Format event of the repeating section so that they will be instantiated only once and use less memory.
- **CacheToDisk** Set the CacheToDisk property of the Document object to True. Although it will slow down the processing time, this will cause the document to be cached to disk instead of loading the whole report in memory. The PDF export will also detect this setting and use the cached report to export. Please note that only the PDF export is affected by the CacheToDisk Property; other exports may run out of memory with very large reports. Also note that

CacheToDisk uses IsolatedStorage to store a page's canvasItems to disk. To use CacheToDisk you must have IsolatedStorageFilePermission.

- **Summaries** Placing summaries (primarily page count and report totals) in header sections will have an adverse effect on memory as well as rendering speed with large reports using the CacheToDisk property. Since the rendering of the header is delayed until ActiveReports determines the total or page count of the following sections, CacheToDisk is unable to perform any optimization. The greater the number of affected sections, the longer rendering is delayed and the less optimization CacheToDisk will offer. Therefore, a group total in a group header section does not affect performance and memory as much as a report total in the report header.
- **Releasing Reports** To properly release a report instance from memory, take these steps in the following order:
 1. Call the Dispose() method of the Document object
 2. Call the Dispose() method of the Report object
 3. Set the Report object to null

The following code uses the above steps to release a report.

To write the code in Visual Basic.NET

Visual Basic.NET code.

```
rpt.Document.Dispose()  
rpt.Dispose()  
rpt = Nothing
```

To write the code in C#

C# code.


```
rpt.Document.Dispose();  
rpt.Dispose();  
rpt = null;
```

Speed Considerations


- **Images** An image repeated on every page of a very long report is stored only once to improve memory, but the comparison necessary to detect duplicate images slows performance. This is not only the case with the report document itself, but also with the Excel, PDF, and HTML exports as they perform their own comparisons.
- **Summaries** Placing summaries (primarily page count and report totals) in header sections will slow report processing. ActiveReports must determine the total or page count of the following sections before it can render the header section. The greater the number of affected sections, the longer rendering is delayed. Therefore, a group total in a group header section does not affect performance and memory as much as a report total in the report header.
- **CacheToDisk** Be sure that the CacheToDisk property of the Document object is not set to True. Setting it to True increases the amount of time the report takes to load, and should only be used with very large reports that use a lot of memory. If this is used with smaller reports of less than 100 pages, it may actually cause more memory to be used.
- **Stored Procedures** Using stored procedures instead of SELECT statements speeds up the processing time of a report, as it reduces the number of records handled by ActiveReports. Using SELECT * statements is not recommended unless you are actually using all of the data returned by such a statement. Consult your database administrator for other ways to speed up data retrieval, such as indexing tables.

CacheToDisk and Resource Storage

There are several internal settings to consider when you use the CacheToDisk property of the ActiveReports Document object.

 **Note:** You must have IsolatedStorageFilePermission in order to use the CacheToDisk property. The use of CacheToDisk slows processing time.

The **CacheToDisk** property tells ActiveReports whether to move report resources to IsolatedStorage instead of holding it in memory. With this property set to **False**, all memory is used to store resources. If the **CacheToDisk** property is **True** and the **CacheToDiskLocation** property is not set, the default location in which it caches resources is IsolatedStorage, so you must have IsolatedStorageFilePermission in order to use it. The cache capacity for IsolatedStorage may depend on your configuration, but does not exceed 3 GB.

 **Note:** Temporary files and folders created in IsolatedStorage are not deleted automatically.

To avoid using IsolatedStorage, you can specify a folder in the **CacheToDiskLocation** property. For an example of the code used to turn on CacheToDisk and specify a folder, see the **CacheToDiskLocation ('CacheToDiskLocation Property' in the on-line documentation)** property in the Class Library documentation.

Section 508 Compliance

Section 508 requires that when Federal agencies develop, procure, maintain, or use electronic and information technology, Federal employees with disabilities have access to and use of information and data that is comparable to the access and use by Federal employees without disabilities, unless an undue burden would be imposed on the agency. Section 508 also requires that individuals with disabilities seeking information or services from a Federal agency have access to and use of information and data that is comparable to that provided to the general public, unless an undue burden would be imposed on the agency.

Accessibility Summary:

All major features of ActiveReports software are accessible via keyboard navigation.

DISCLAIMER:

GRAPECITY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. The following information reflects the general accessibility features of GrapeCity software components as related to the Section 508 standards. If you find that the information is not accurate, or if you have specific accessibility needs that our products do not meet, please contact us and we will attempt to rectify the problem, although we cannot guarantee that we will be able to do so in every case.

ActiveReports Viewer and End User Designer controls

Section 1194.21 Software applications and operating systems

Criteria

(a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

(b) Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the

Status

Supported

Supported

Remarks

Each of the toolboxes, toolbox items, toolbars, buttons, menu items, and context menus is executable from a keyboard, and the functions performed by each can be discerned textually. Hyperlinks supported within the control can also provide textual cues provided by the developer.

The controls do not disrupt or disable industry standard accessibility features of other products.

application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.

(c) A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that Assistive Technology can track focus and focus changes.

Supported Focus on interface elements is programmatically exposed so that Assistive Technology can track focus and focus changes.

(d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to Assistive Technology. When an image represents a program element, the information conveyed by the image must also be available in text.

Supported Each of the user interface elements provides Assistive Technology with information about its identity, operation, and state. Any images representing a user interface element also provide a textual conveyance of the information.

(e) When bitmap images are used to identify controls, status indicators, or other programmatic elements, the meaning assigned to those images shall be consistent throughout an application's performance.

Supported Any images used to identify a programmatic element have a consistent meaning throughout the controls.

(f) Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

Supported Textual information about all viewer elements includes text content, caret location, and any text attributes (i.e. bold, italic, size, color, etc.).

(g) Applications shall not override user selected contrast and color selections and other individual display attributes.

Supported The controls do not, by default, override local settings for contrast and color or other display attributes.

(h) When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

Not Applicable There are no animated elements in the controls, so a non-animated presentation mode is not necessary.

(i) Color coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

Supported Any interface element that uses color to indicate an action, prompt a response, or distinguish a visual element also provides a textual cue.

(j) When a product permits a user to adjust color and contrast settings, a variety of color selections capable of producing a range of contrast levels shall be provided.

Supported By default, no color or contrast settings are available to the user unless the developer adds them; therefore it is the responsibility of the developer to provide the user with a variety of color selections, if applicable, so that they can produce a range of contrast levels.

(k) Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.

(l) When electronic forms are used, the form shall allow people using Assistive Technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

Supported The controls do not use flashing or blinking text or objects.

Supported Any form-type dialogs or windows associated with the controls provide Assistive Technology with access to information on all directions, cues, field elements, and functionality required for completion.

Web Viewer control (Professional Edition)

Section 1194.22 Web-based Internet information and applications

Criteria

(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

(b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

(d) Documents shall be organized so they are readable without requiring an associated style sheet.

(e) Redundant text links shall be provided for each active region of a server-side image map.

(f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

Status Remarks

Supported Each non-text element has a text equivalent.

Not Applicable There is no multimedia presentation associated with the software.

Not Applicable By default, no information is conveyed with color.

Supported ActiveReports provides a text export, so that any generated reports can be issued as plain text.

Not Applicable There are no server-side image maps associated with the software.

Not Applicable There are no client-side image maps associated with the software.

Not Applicable By default, there are no data tables associated with the software.

Not Applicable By default, there are no data tables associated with the software.

| | | |
|--|----------------|--|
| (i) Frames shall be titled with text that facilitates frame identification and navigation. | Not Applicable | By default, the software does not use frames. |
| (j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz. | Supported | The software does not cause the screen to flicker outside the recommended range. |
| (k) A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes. | Supported | ActiveReports provides a text export, so that any generated reports can be issued as plain text. |
| (l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by Assistive Technology. | Supported | ActiveReports provides alternatives to using scripting languages in the WebViewer control. |
| (m) When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l). | Supported | The ActiveReports WebViewer has a Flash ViewerType that includes a link to the plug-in. |
| (n) When electronic forms are designed to be completed on-line, the form shall allow people using Assistive Technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues. | Not Applicable | There are no electronic forms associated with ActiveReports. |
| (o) A method shall be provided that permits users to skip repetitive navigation links. | Not Applicable | By default, the software contains no repetitive navigation links. |
| (p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required. | Not Applicable | No timed responses are required with ActiveReports. |

ActiveReports

Section 1194.31 Functional Performance Criteria

| Criteria | Status | Remarks |
|--|---------------|---------|
| (a) At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided. | Not Supported | |
| (b) At least one mode of operation and information retrieval that does not require visual acuity greater than 20/70 shall be provided in audio and enlarged print output working together or independently, or support for assistive technology used by people who are | Not Supported | |

visually impaired shall be provided.

(c) At least one mode of operation and information retrieval that does not require user hearing shall be provided, or support for assistive technology used by people who are deaf or hard of hearing shall be provided. Not Applicable

(d) Where audio information is important for the use of a product, at least one mode of operation and information retrieval shall be provided in an enhanced auditory fashion, or support for assistive hearing devices shall be provided. Not Applicable

(e) At least one mode of operation and information retrieval that does not require user speech shall be provided, or support for assistive technology used by people with disabilities shall be provided. Not Applicable

(f) At least one mode of operation and information retrieval that does not require fine motor control or simultaneous actions and that is operable with limited reach and strength shall be provided. Not Applicable

Documentation

Section 1194.41 Information, Documentation, and Support

| Criteria | Status | Remarks |
|---|---------------------------|---|
| (a) Product support documentation provided to end-users shall be made available in alternate formats upon request, at no additional charge. | Supported | Documentation is available in three formats: hxs (Visual Studio Integrated help), chm, and pdf. |
| (b) End-users shall have access to a description of the accessibility and compatibility features of products in alternate formats or alternate methods upon request, at no additional charge. | Supported | Accessibility information is available upon request. |
| (c) Support services for products shall accommodate the communication needs of end-users with disabilities. | Supported with Exceptions | Support services are available by telephone and by email and forum. |

Localization

ActiveReports uses the Hub and Spoke model for localizing resources. The hub is the main executing assembly, for example, the Viewer Control, ActiveReports.Viewer6.dll.

The spokes are the satellite DLLs that contain localized resources for the application, for example, ActiveReports.Viewer6.resources.dll.


The Localization folder, **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, the localization folder is **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**), contains everything you need to localize all of your ActiveReports components.

Within that folder, each component of ActiveReports that you can localize, 14 in all, has two files:

- ***.bat** Set the **culture** to which you want to localize.
- ***.zip** Change the strings in the resource files (*.resx) it contains.

There is one application in the folder: **NameCompleter.exe** When you run your bat file after changing it to your culture, it runs this application to create a SatelliteAssembly folder with a language subfolder containing the localized ActiveReports.AssemblyName.resources.dll file.

Place the **culture** subdirectories containing the satellite assemblies in the folder that contains your main executing assembly.

 **Note:** If you want to put your localization in the Global Assembly Cache (GAC), you must first send the localized `ActiveReports.AssemblyName.resources.dll` file to [GrapeCity](#) and get it signed. Then you can drag the language subfolder with the signed dll file into `C:\WINDOWS\ASSEMBLY`.

When the main executing assembly needs a resource, it uses a `ResourceManager` object to load the required resource. The `ResourceManager` uses the thread's `CurrentUICulture` property.


The common language runtime sets the `CurrentUICulture` property or you can set it in code to force a certain UI Culture to test whether your satellite DLL is loading properly. The `ResourceManager` class uses the `CurrentUICulture` Property to locate subdirectories that contain a satellite DLL for the current culture. If no subdirectory exists, the `ResourceManager` uses the resource that is embedded in the assembly. US English is the default culture for ActiveReports.

 For more detailed information about how the Framework locates satellite DLLs, please refer to the help system in Visual Studio® or the book *Developing International Software, 2nd edition* by MS Press that contains information on localizing applications using the .NET Framework.

Designer Control (Pro Edition)

With the Professional Edition of ActiveReports, you can host the ActiveReports Designer control in your Windows Forms application and provide your end users with report editing capabilities. The control's methods and properties allow you to save and load report layouts, monitor and control the design environment, and customize the look and feel.

In addition to the Designer control, ActiveReports offers a `CreateToolStrips` method to help you add default toolbars to the designer and add and remove individual tool bars and commands. This gives your designer a finished look and allows you to quickly create a functioning report designer application.

 **Note:** You cannot host the ActiveReports Designer control in the Web application and Web site project types.

See the **Add Designer ToolStrips** topic for more information.

How To

See step-by-step instructions for performing common tasks using ActiveReports.

This section contains information about how to:

Work with Data

Learn to bind reports to data, group data, and modify data sources at run time.

Work with Fields

Learn how to add field expressions and to create summary fields and calculated fields.

Create Common Reports

Learn how to create top N, summary, and green bar reports.

Change Ruler Measurements

Learn how to change the units associated with the ruler from standard to metric.

Display Page Numbers and Report Dates

Learn how to quickly add Page N of M and report dates and times to your reports.

Use XML Data with Barcodes

Learn how to set the data type of XML data source fields so that the Barcode control can read them.

Add Hyperlinks

Learn how hyperlinks can be used in ActiveReports.

Add Annotations

Learn how to add text, shapes, arrows, and lines to your reports as temporary or persistent annotations.

Export Reports

Learn to export reports to each of the supported formats.

Print Multiple Copies, Duplex, and Landscape

Learn to use the Printer Settings tab of the Report Settings window.

Conditionally Show or Hide Details

Learn to prevent rendering of the detail section for data that meets certain conditions.

Use External Style Sheets

Learn to use custom styles, save them externally, and use them in other reports.

Add Bookmarks

Learn how to use bookmarks in ActiveReports.

Insert or Add Pages

Learn how to add or insert specified pages from one report into another.

Create Charts

Learn to create charts using the Chart control.

Load a File into a RichText Control

Learn to load an HTML or RTF file into the RichText control.

Use Custom Controls on Reports (TreeView)

Learn to access custom or third-party controls in ActiveReports code.

Create Report Templates (Inheritance)

Learn how to use Inheritance to create report templates.

Add Parameters

Learn to pass parameters into reports in several ways.

Embed Subreports in a Report

Learn to use the Subreport control.

Pass Parameters to a Subreport

Learn to pass parameters from a parent report to a subreport.

Save and Load Report Files (RDF)

Learn how to save and load reports as RDF files at run time.

Save and Load Report Layout Files (RPX)

Learn how to save and load reports as RPX-based report layouts at run time.

Add Code to Layouts Using Script

Learn the specifics of using the scripting capabilities of ActiveReports for .NET in your applications.

Provide One-Touch Printing in the WebViewer (Pro Edition)

Learn how to set up one-touch printing with the new FlashViewer.

Add Designer ToolStrips

Learn how to add the designer control's ToolStrips to a ToolStripContainer.

Configure HTTPHandlers (Pro Edition)

Learn to configure HttpHandlers in IIS so that you can use them to display ActiveReports on the Web.

Add Report Links to Web Forms (Pro Edition)

Learn to display ActiveReports from hyperlinks that include parameter values or specify output formats.

Provide PDF Printing in Silverlight (Pro Edition)

Learn how you can print a report from the Silverlight Viewer directly to the PDF format.

Use the Silverlight Viewer in Silverlight Out-of-Browser Applications (Pro Edition)

Learn how to configure your Silverlight-based application to run outside a browser.

Make Changes to a Report Undoable in the End User Designer (Pro Edition)

Learn how to make undoable changes in the properties of the ActiveReports report, section and control classes in code-behind.

Customize, Localize and Deploy

Learn to localize the components of ActiveReports, and how to deploy your applications.

Work with Data

See step-by-step instructions for performing common tasks using ActiveReports.

This section contains information about how to:**Bind Reports to a Data Source**

Learn how to bind reports to various data sources, datasets, and data views.

Group Data

Learn how to use the GroupHeader section to group data in a report.

Modify Data Sources at Run Time


Learn to use code to modify a report's data source.

Bind Reports to a Data Source

Add data at design time using the Report Explorer.

To add Calculated Fields

1. In the Report Explorer, expand the **Fields** node.
2. Right-click the **Calculated** node and select **Add**.
3. With the field selected in the Properties window, enter a **Formula** for the calculation (for example, `UnitPrice * Discount`).

 **Note:** Do not use an equals sign (=) at the beginning of the formula as you would in the **DataField** property of a textbox.

4. Drag the calculated field onto the design surface of the report to create a bound textbox.

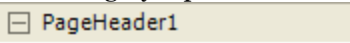
To add Parameters

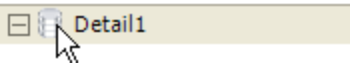
1. In the Report Explorer, right-click the **Parameters** node and select **Add**.
2. With the parameter selected in the Properties window, enter a **DefaultValue** for the parameter.
3. Select whether to **PromptUser** for a parameter value, and if so, supply a string for the **Prompt**.

Bind a report to a data source using the data source icon in the detail section band which opens the Report Data Source window. There are four tabs on the window for the four most commonly used data sources:

To use the OLE DB data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

 PageHeader1

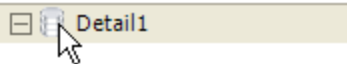
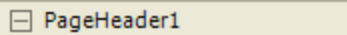
 Detail1

2. On the **OLE DB** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.

4. Click the ellipsis (...) button to browse to your database or the sample Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter a SQL query to select the data that you want.
7. Click **OK** to save the data source and return to the report design surface.

To use the SQL data source

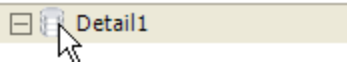
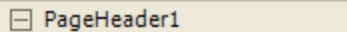
1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



2. On the **SQL** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft OLE DB Provider for SQL Server** and click the **Next** button.
4. Under **1. Select or enter a server name**, drop down the box and select your server.
5. Under **2. Enter information to log on to the server**, set up your log on information.
6. Under **3.** you can select a database on the server or attach a database file.
7. Click **OK** to close the window and fill in the Connection String field.
8. In the Query field, enter a SQL query to select the data that you want.
9. Click **OK** to save the data source and return to the report design surface.

To use the XML data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



2. On the **XML** tab, next to File URL, click the **..** button.
3. In the Open File window that appears, navigate to your XML data file, select it, and click the **Open** button (the sample xml data file is located in C:\Program Files\GrapeCity\ActiveReports 6\Data\customer.xml; on a **64-bit Windows operating system** the file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\customer.xml).
4. In the Recordset Pattern field, enter a valid XPath expression. (for example, `//CUSTOMER` for the sample xml data file)
5. Click **OK** to save the data source and return to the report design surface.

To use an Unbound data source

1. Double-click in the gray area below the design area of your report to create an event-handling method for the **ReportStart** event.
2. Add code to:
 - Change the data source at run time
 - Close the data connection
 - Add fields to the report
 - Populate fields in the report

To create a data source in Visual Basic.NET

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

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim conn As System.Data.OleDb.OleDbConnection
Dim reader As System.Data.OleDb.OleDbDataReader
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Dim dbPath As String = getDatabasePath()
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb"
conn = New System.Data.OleDb.OleDbConnection(connString)
Dim cmdText As String = "SELECT Categories.*, Products.* FROM Products INNER JOIN
Categories ON Categories.CategoryID = Products.CategoryID WHERE Products.UnitPrice =
18"
Dim cmd As New System.Data.OleDb.OleDbCommand(cmdText, conn)
conn.Open()
reader = cmd.ExecuteReader()
```

To create a data source in C#

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

C# code. Paste JUST ABOVE the ReportStart event.

```
private static System.Data.OleDb.OleDbConnection conn;
private static System.Data.OleDb.OleDbDataReader reader;
```

C# code. Paste INSIDE the ReportStart event.

```
string dbPath = getDatabasePath();
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb";
conn = new System.Data.OleDb.OleDbConnection(connString);
string cmdText = "SELECT Categories.*, Products.* FROM Products INNER JOIN Categories
ON Categories.CategoryID = Products.CategoryID WHERE Products.UnitPrice = 18";
System.Data.OleDb.OleDbCommand cmd = new System.Data.OleDb.OleDbCommand(cmdText, conn);
conn.Open();
reader = cmd.ExecuteReader();
```

To close the data connection in Visual Basic

1. In design view of YourReportName, drop down the field at the top left of the code view and select **(YourReportName Events)**.
2. Drop down the field at the top right of the code view and select **ReportEnd**. This creates an event-handling method for ReportEnd event.
3. Add code to the handler to close the data connection.

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

Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
reader.Close()
conn.Close()
```

To close the data connection in C#


1. Click in the gray area below YourReportName to select the report.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportEnd**. This creates an event-handling method for the ReportEnd event.

4. Add code to the handler to close the data connection.

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

C# code. Paste INSIDE the ReportEnd event.

```
reader.Close();  
conn.Close();
```

 **Warning:** Do not access the Fields collection outside the **DataInitialize** and **FetchData** events. Accessing the Fields collection outside of these events is not supported, and has unpredictable results.

To add fields in Visual Basic

1. Right-click in any section of the design surface of the report, and select **View Code** to display the code view for the report.
2. At the top left of the code view of the report, click the drop-down arrow and select **(YourReportName Events)**.
3. At the top right of the code window, click the drop-down arrow and select **DataInitialize**. This creates an event-handling method for the report's DataInitialize event.
4. 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.NET code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryID")  
Fields.Add("CategoryName")  
Fields.Add("ProductName")  
Fields.Add("UnitsInStock")  
Fields.Add("Description")  
Fields.Add("TotalLabel")
```

To add fields in C#

1. Click in the gray area below the report to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **DataInitialize**. This creates an event-handling method for the report's DataInitialize event.
4. 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.

C# code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryID");  
Fields.Add("CategoryName");  
Fields.Add("ProductName");  
Fields.Add("UnitsInStock");  
Fields.Add("Description");  
Fields.Add("TotalLabel");
```

To populate fields in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(YourReportName Events)**.
2. At the top right of the code window, click the drop-down arrow and select **FetchData**. This creates an event-handling method for the report's FetchData event.
3. 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.NET code. Paste INSIDE the FetchData event.

```

Try
    reader.Read()
    Me.Fields("CategoryID").Value = reader("categories.CategoryID")
    Me.Fields("CategoryName").Value = reader("CategoryName")
    Me.Fields("ProductName").Value = reader("ProductName")
    Me.Fields("UnitsInStock").Value = reader("UnitsInStock")
    Me.Fields("Description").Value = reader("Description")
    Me.Fields("TotalLabel").Value = "Total Number of " + reader("CategoryName") + ":"
    eArgs.EOF = False
Catch
    eArgs.EOF = True
End Try

```

To populate fields in C#

1. Back in design view, click in the gray area below the report to select it.
2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
4. 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.

C# code. Paste INSIDE the FetchData event.

```

try
{
    reader.Read();
    Fields["CategoryID"].Value = reader["categories.CategoryID"].ToString();
    Fields["CategoryName"].Value = reader["CategoryName"].ToString();
    Fields["ProductName"].Value = reader["ProductName"].ToString();
    Fields["UnitsInStock"].Value = reader["UnitsInStock"].ToString();
    Fields["Description"].Value = reader["Description"].ToString();
    Fields["TotalLabel"].Value = "Total Number of " +
reader["CategoryName"].ToString() + ":";
    eArgs.EOF = false;
}
catch
{
    eArgs.EOF = true;
}

```

To use the IEnumerable data source

1. Double-click in the gray area below the design area of your report to display the code view for the report.
2. Add the following code as shown in the examples below.

To create a data source in Visual Basic**Paste INSIDE the class declaration of the report**

```

Private datasource1 As IEnumerable(Of String) = Nothing
Dim list As List(Of String) = Nothing

```

Paste INSIDE the class declaration of the report

```

Private Function GetIEnumerableData() As IEnumerable(Of String)
    For i As Integer = 1 To 10
        list.Add(String.Format("TestData_{0}", i.ToString()))
    Next

```

```
Return list
End Function
```

To create a data source in C#

Paste JUST ABOVE the InitializeComponent method

```
private IEnumerator<string> datasource = null;
```

Paste BELOW the InitializeComponent method

```
private IEnumerable<string> GetIEnumerableData()
{
    for (int i = 1; i <= 10; i++)
    {
        yield return string.Format("TestData_{0}", i.ToString());
    }
}
```

3. Back in design view, click in the gray area below the report to select it.
4. Click the events icon in the Properties window to display available events for the report.
5. Double-click **DataInitialize**. This creates an event-handling method for the report's DataInitialize event.
6. Add code to the handler to add fields to the report's Fields collection.

To add fields in Visual Basic

Paste INSIDE the DataInitialize event

```
Me.Fields.Add("TestField")
Me.list = New List(Of String)
datasource1 = GetIEnumerableData().GetEnumerator()
```

To add fields in C#

Paste INSIDE the DataInitialize event

```
this.Fields.Add("TestField");
datasource = GetIEnumerableData().GetEnumerator();
```

7. Back in design view, click in the gray area below the report to select it.
8. Click the events icon in the Properties window to display available events for the report.
9. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
10. Add code to the handler to retrieve information to populate the report fields.

To populate fields in Visual Basic

Paste INSIDE the FetchData event

```
If datasource1.MoveNext() Then
    Me.Fields("TestField").Value = datasource1.Current
    eArgs.EOF = False
Else
    eArgs.EOF = True
End If
```

To populate fields in C#

Paste INSIDE the FetchData event

```
if (datasource.MoveNext())
{
    this.Fields["TestField"].Value = datasource.Current;
```

```
eArgs.EOF = false;
}
else
    eArgs.EOF = true;
```

Group Data

To group a report on a field

1. Right-click the design surface of a report and select **Insert**, then **Group Header/Footer** to add a group header and group footer section.
2. With the group header selected in the Properties window, drop down the **DataField** property and select the field on which to group the report.

To group a report on a field expression

1. Right-click the design surface of a report and select **Insert**, then **Group Header/Footer** to add a group header and group footer section.
2. Enter a field expression in the **DataField** property, for example, **=City + Country** (For more information on field expressions, see the **Add Field Expressions** topic.)

Modify Data Sources at Run Time

ActiveReports allows you to modify your data source at run time. Below is sample code that you can use to connect a report to the Nwind.mdb sample database at run time.

To find the database path

1. Right-click in any section of the design window of the report, and select **View Code** to display the code view for the report.
2. Add code to the report to get the sample database path from the registry.

To write the code in Visual Basic

The following example shows what the code for the function looks like.

Visual Basic.NET code. Paste JUST BELOW the Imports DataDynamics.ActiveReports statements at the top of the code view.

```
Imports System
Imports Microsoft.Win32
```

Visual Basic.NET code. Paste INSIDE the report class and hit ENTER.

```
Private Function getDatabasePath() As String
```

This creates a function for getDatabasePath.

Visual Basic.NET code. Paste INSIDE the getDatabasePath function.

```
Dim regKey As RegistryKey
regKey = Registry.LocalMachine
regKey = regKey.CreateSubKey("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB")
getDatabasePath = CType(regKey.GetValue(""), String)
```

To write the code in C#

C# code. Paste JUST BELOW the using DataDynamics.ActiveReports; statements at the top of the code view.

```
using Microsoft.Win32;
using System;
```

C# code. Paste INSIDE the report class and hit ENTER.

```
private string getDatabasePath()
```

This creates a function for getDatabasePath.

C# code. Paste INSIDE the getDatabasePath function.

```
RegistryKey regKey = Registry.LocalMachine;
regKey = regKey.CreateSubKey("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB");
return ((string) (regKey.GetValue("")));
```

To change the data source at run time

1. Double-click in the gray area below the report to create an event-handling method for the **ReportStart** event.
2. Add code to the handler to change the data source at run time.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim conn As System.Data.OleDb.OleDbConnection
Dim reader As System.Data.OleDb.OleDbDataReader
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Dim dbPath As String = getDatabasePath()
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb"
conn = New System.Data.OleDb.OleDbConnection(connString)
Dim cmd As New System.Data.OleDb.OleDbCommand("SELECT * FROM Products WHERE UnitPrice =
18", conn)
conn.Open()
reader = cmd.ExecuteReader()
Me.DataSource = reader
```

To write the code in C#

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

C# code. Paste JUST ABOVE the ReportStart event.

```
private static System.Data.OleDb.OleDbConnection conn;
private static System.Data.OleDb.OleDbDataReader reader;
```

C# code. Paste INSIDE the ReportStart event.

```
string dbPath = getDatabasePath();
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb";
```

```
conn = new System.Data.OleDb.OleDbConnection(connString);
System.Data.OleDb.OleDbCommand cmd = new System.Data.OleDb.OleDbCommand("SELECT * FROM
Products WHERE UnitPrice = 18", conn);
conn.Open();
reader = cmd.ExecuteReader();
this.DataSource = reader;
```

To close the data connection

To write the code in Visual Basic

1. In design view of rptModifyDS, drop down the field at the top left of the code view and select **(rptYourReport Events)**.
2. Drop down the field at the top right of the code view and select **ReportEnd**. This creates an event-handling method for ReportEnd event.
3. Add code to the handler to close the data connection.

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

Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
reader.Close()
conn.Close()
```

To write the code in C#

1. Click in the gray area below rptModifyDS to select the report.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportEnd**. This creates an event-handling method for the ReportEnd event.
4. Add code to the handler to close the data connection.

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

C# code. Paste INSIDE the ReportEnd event.

```
reader.Close();
conn.Close();
```

Work with Fields

See step-by-step instructions for using properties to manipulate fields.

This section contains information about how to:

Add Field Expressions

Learn how to add field expressions to a text box data field.

Create Summary Fields

Learn how to create subtotal and grand total fields.

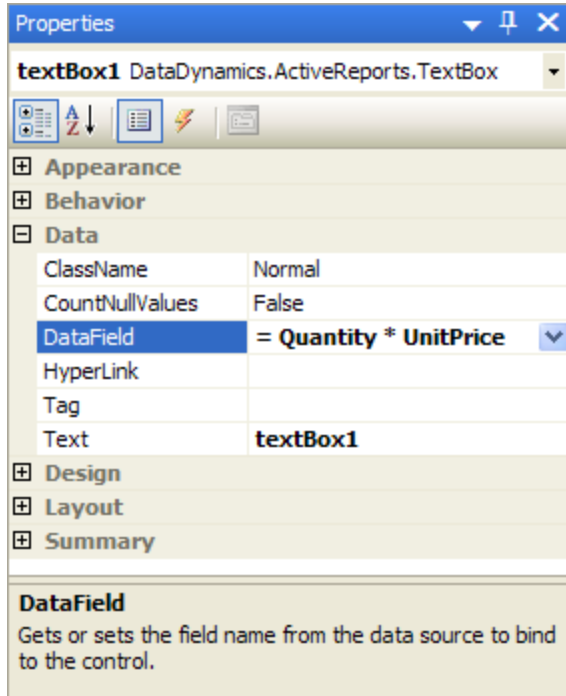
Create Calculated Fields

Learn how to add field expressions to a text box data field.

Add Field Expressions

In ActiveReports, C# expressions can be used in the DataField property to specify textbox output in a report, such as date/time, mathematical calculations or conditional values. All field expressions used in the DataField property begin

with the equals sign (=).



Using Field Expressions

To use a mathematical expression

Change the `DataField` property for the text box to the mathematical calculation desired.

Examples: `=UnitPrice+5`

`=Quantity-5`

`=Quantity*UnitPrice`

`=UnitPrice/QuantityPerUnit`

To use a substring

Change the `DataField` property for the text box to the substring needed. If setting up grouping, change the `GroupHeader`'s `DataField` property to the same substring.

Example: `=ProductName.Substring(0, 1)`

To use date/time

Change the `DataField` property for the text box to the following.

Example: `=System.DateTime.Now.ToString()`

To create a conditional value

Change the `DataField` property for the text box to the conditional statement desired.

Example: `=(UnitsInStock > 0)?"In Stock":"Backorder"`

To concatenate fields

Change the DataField property for the text box to the following.

```
Examples: ="There are " + UnitsInStock + " units of " + ProductName + " in stock."  
         =TitleOfCourtesy + " " + FirstName + " " + LastName
```



Note: ActiveReports automatically handles null values, replacing them with an empty string.

To round a calculation

Change the DataField Property for the text box to the following.

```
Example: =(double)System.Math.Round(UnitPrice*UnitsOnOrder,2)
```

To use modular division

Change the DataField Property for the text box to the following to get the remainder (2 in this case).

```
Example: =22%(5)
```

To replace a null value

Change the DataField Property for the text box to the following to replace nulls with your own value.

```
=(Region == System.DBNull.Value) ? "No region specified" : Region
```

Create 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.

To add a group summary field

1. Right-click the design surface of the report and select **Insert**, then **Group Header/Footer** to add a group header and group footer section to the report.
2. With the group header selected in the Properties window, change the **DataField** property to the field on which to group the report.
3. Drag a field that you want to summarize onto the group footer section and set its properties as follows:
 - SummaryType = SubTotal
 - SummaryRunning = Group
 - SummaryGroup = *YourGroupHeaderName*

To add a report summary field

1. Right-click the design surface of the report and select **Insert**, then **Report Header/Footer** to add a report header and group footer section to the report.
2. Drag a field that you want to summarize onto the report footer section and set its properties as follows:
 - SummaryType = GrandTotal
 - SummaryRunning = All

 **Note:** The **SummaryRunning** property is only set when the SummaryType is GrandTotal or SubTotal, otherwise it is set to None.

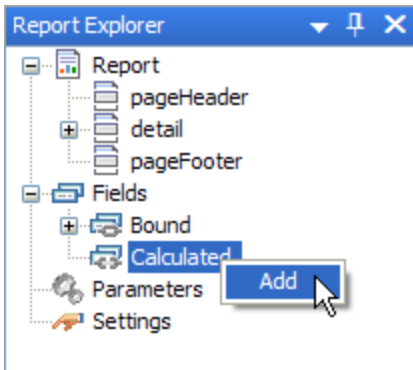
Distinct summarization can be used in a situation when the field's value repeats in several detail records and the summary function needs to include a single value from all repeating values. To do this, you would need to set the **DistinctField** property of the summary field to the appropriate value and set the **SummaryFunc** property to the appropriate distinct summary function (for example, DSum for distinct summary or DCount for distinct count).

Create Calculated Fields

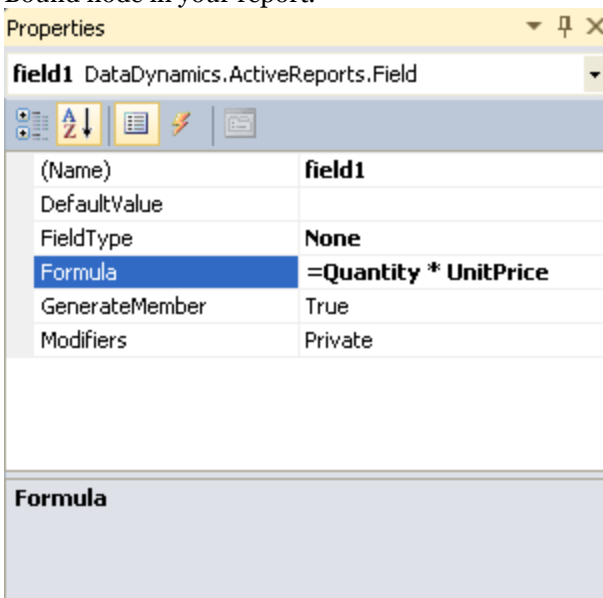
ActiveReports allows you to add calculated fields to a report using the Report Explorer, or you can use the DataField property of any textbox to perform calculations based on the value of specific data fields. This how-to topic multiplies the values of two bound fields, but you can use any valid Field Expression operators. See the **Add Field Expressions** topic for more information.

To create a calculated field

1. In the Report Explorer, expand the **Fields** node, right-click on the **Calculated** node, and select **Add**. This creates an unbound field named "field1" that can be used to perform custom calculations.



2. With field1 selected in the Properties Window, change the **Formula** property to **=Quantity * UnitPrice** to bind it to the product of the Quantity and UnitPrice fields, substituting the names of fields that you have under the Bound node in your report.



3. Change other properties as desired.

4. Drag the field from the Calculated node onto the detail section of the report. This creates an ActiveReports TextBox object, and sets its DataField property to the name of the calculated field.

Create Common Reports

See step-by-step instructions for creating commonly used reports with ActiveReports.

This section contains information about how to:

Create Top N Reports

Learn to get top 10 data from an Access database and display the data in fields.

Create Summary Reports

Learn to display summary data while hiding detail.

Create Green Bar Reports

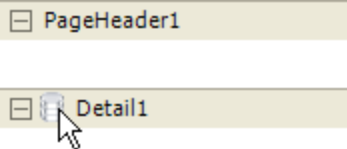
Learn to alternate background colors on the detail section.

Create Top N Reports

To display only the top ten (or other number) of details on a report, you can manipulate the data pulled by your SQL query.

To set an Access data source to pull top ten data

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



2. On the **OLE DB** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, paste the following SQL query.

SQL Query

```
SELECT TOP 10 Customers.CompanyName, Sum([UnitPrice]*[Quantity])
AS Sales
FROM (Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID)
INNER JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
GROUP BY Customers.CompanyName
ORDER BY Sum([UnitPrice]*[Quantity])
DESC
```


7. Click **OK** to return to the report design surface.

To add controls to display the top ten data

1. In the Report Explorer, expand the **Fields** node, then the **Bound** node.

2. Drag the following fields onto the detail section and set the properties of each textbox as indicated.

| Field | Text | Location | Miscellaneous |
|--------------|--------------|-----------------|-------------------------|
| CompanyName | Company Name | 0.5, 0 | |
| Sales | Sales | 5, 0 | OutputFormat = Currency |

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

Create Summary Reports

To show only summary data in a report, set the Detail section's Visible property to False or set its Height to 0. The Detail section still processes, but only the the summary fields in the GroupHeader and Footer sections are displayed.

To create a summary report

1. With the Detail section selected in the Properties window, set the **Visible** property to **False**.
2. Right-click the design surface of a report and select **Insert**, then **Group Header/Footer** to add a group header and group footer section.
3. With the group header selected in the Properties window, drop down the **DataField** property and select the field on which to group the report.
4. Add a TextBox control to the GroupFooter section, and set its properties as follows:
 - **DataField:** the field that you want to summarize
 - **SummaryGroup:** the name of the GroupHeader section
 - **SummaryRunning:** Group
 - **SummaryType:** SubTotal

To display a grand total at the end of the report

1. Right-click the design surface of a report and select **Insert**, then **Report Header/Footer** to add a report header and report footer section.
2. Add a TextBox control to the ReportFooter section, and set its properties as follows:
 - **DataField:** the field that you want to summarize
 - **SummaryRunning:** All
 - **SummaryType:** GrandTotal

Create 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.

| <u>Product ID</u> | <u>Product Name</u> | <u>Units in Stock</u> | <u>Units on</u> |
|-------------------|---------------------------------|-----------------------|-----------------|
| 1 | Chai | 39 | |
| 2 | Chang | 17 | |
| 3 | Aniseed Syrup | 13 | |
| 4 | Chef Anton's Cajun Seasoning | 53 | |
| 5 | Chef Anton's Gumbo Mix | 0 | |
| 6 | Grandma's Boysenberry Spread | 120 | |
| 7 | Uncle Bob's Organic Dried Pears | 15 | |
| 8 | Northwoods Cranberry Sauce | 6 | |
| 9 | Mishi Kobe Niku | 29 | |
| 10 | Ikura | 31 | |

To add code to alternate colors in the detail section

1. Double-click the detail section of the report to create an event-handling method for the Detail Format event.
2. Add code to the handler to alternate background colors.

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste JUST ABOVE the Detail Format event.

```
Dim color As Boolean
```

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
If color = True Then
    Me.Detail1.BackColor = System.Drawing.Color.DarkSeaGreen
    color = False
Else
    Me.Detail1.BackColor = System.Drawing.Color.Transparent
    color = True
End If
```

To write the code in C#

C# code. Paste JUST ABOVE the Detail Format event.

```
bool color;
```

C# code. Paste INSIDE the Detail Format event.

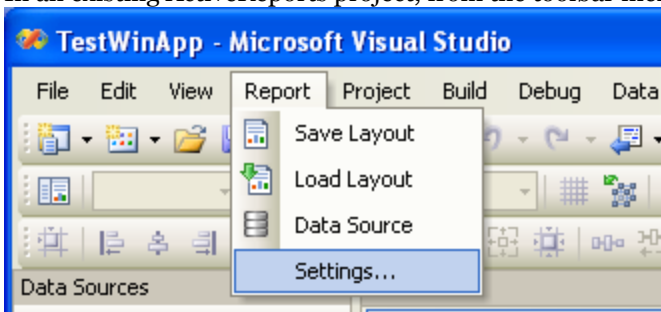
```
if(color)
{
    this.detail.BackColor = System.Drawing.Color.DarkSeaGreen;
    color = false;
}
else
{
    this.detail.BackColor = System.Drawing.Color.Transparent;
    color = true;
}
```

Change Ruler Measurements

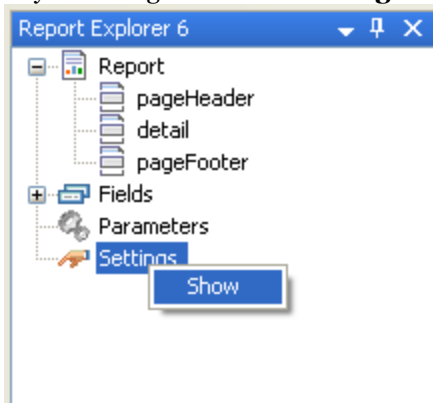
In ActiveReports, ruler measurements can be changed from inches to centimeters and centimeters to inches at both design time and run time.

To change ruler measurements at design time

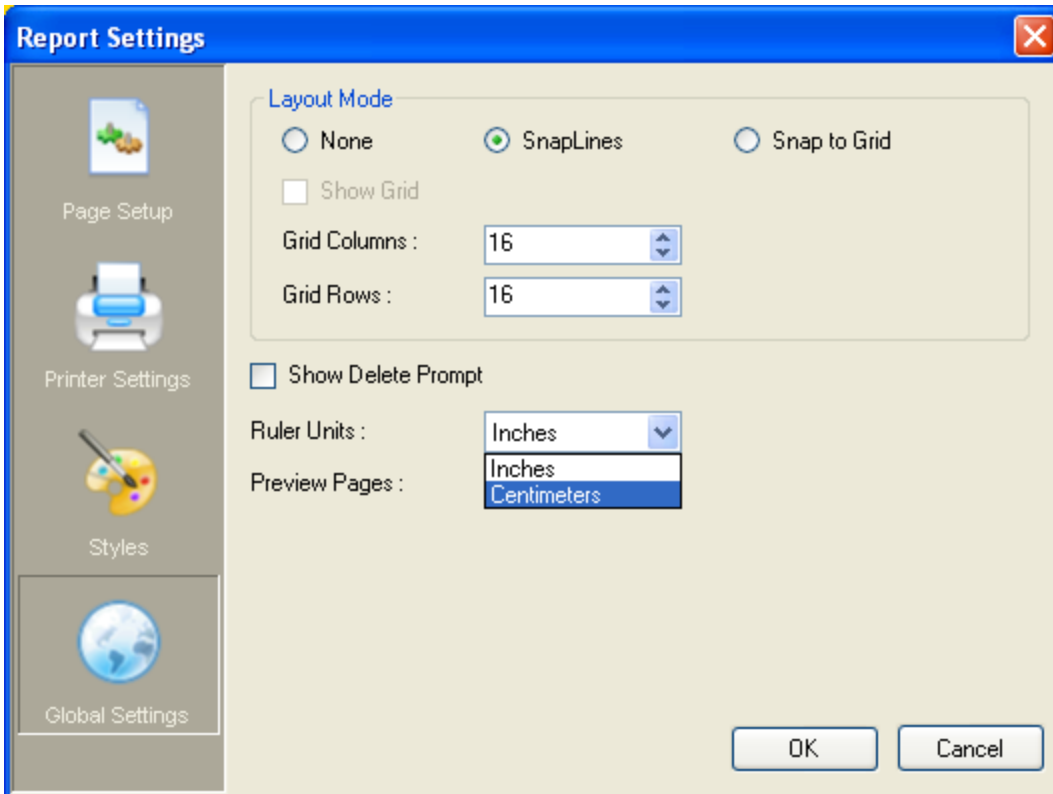
1. In an existing ActiveReports project, from the toolbar menu, select **Report**, then **Settings**.



2. Or you can right-click the **Settings** node in the Report Explorer and select **Show**.



3. In the **Report Settings** dialog, click **Global Settings**.
4. Change **Ruler Units** from inches to centimeters or centimeters to inches.



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.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

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

To write the code in C#

C# code. Paste INSIDE the Format event.


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

Display Page Numbers and Report Dates

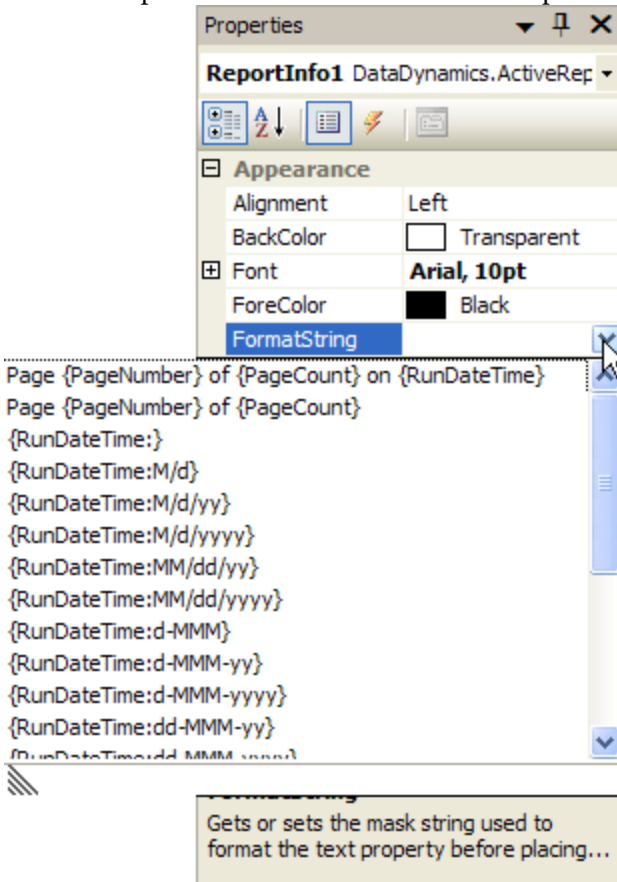
With the ReportInfo control, you can display page numbers and report dates and times by selecting a value in the FormatString property. The page N of M can also be set to a group level using the SummaryGroup and SummaryRunning properties.

To display page numbers and report dates on a report

1. From the ActiveReports toolbox, drag the **ReportInfo** control to the desired location on the report.

 **Note:** With large reports using the CacheToDisk property, placing page counts in header sections may have an adverse effect on memory as well as rendering speed. Since the rendering of the header is delayed until ActiveReports determines the page count of the following sections, CacheToDisk is unable to perform any optimization. For more information on this concept, see **Optimizing ActiveReports**.

2. With the ReportInfo control selected in the Properties Window, drop down the **FormatString** property.



3. Select the preset value that best suits your needs.

You can customize the preset values by editing the string after you select it. For more information on creating formatting strings, see the **Date, Time, and Number Formatting** topic.

To display page numbers and page count at the group level

1. Add a ReportInfo control to the Group Header or Group Footer section of a report and set the **FormatString** property as above.
2. With the ReportInfo control still selected in the Properties Window, drop down the **SummaryGroup** property and select the group for which you want to display a page count.
3. Drop down the **SummaryRunning** property and select **Group**.

Use XML Data with Barcodes

When you use an XML data source to supply data to the Barcode control, the control is unable to automatically detect the data type of the XML node. For this reason, you can use the **DataType** attribute of the XML node to specify string, double, or date time data.



If you have a numeric field with leading zeroes that you need to use with the Barcode control, specify a **string** `DataType` to avoid clipping the zeroes.

To specify a string `DataType` for an XML node and assign it to a barcode

These steps assume that you have already created a **report viewer form** project and added a report (`NewActiveReport1` by default).

1. In the design view of the viewer form, double-click the title bar to open the code view and create a Form Load event.
2. If you are using Visual Basic, paste code like the following above the Form Load event to create a new instance of your report and access its events (in C#, the code is inside the event in the following step):

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the Form Load event.

```
Dim WithEvents rpt As NewActiveReport1
```

3. Paste code like the following inside the Form Load event to create an XML data source, assign it to the report, access the report's `FetchData` event, and display the report in the viewer:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim xdoc As New System.Xml.XmlDocument()
Dim xmlsource As String = "<prj>" + _
    "<bcData>0003456</bcData>" + _
    "<bcData>0003457</bcData>" + _
    "<bcData>0003458</bcData>" + _
    "<bcData>0003459</bcData>" + _
    "</prj>"

rpt = New rptEmpty()
Dim xds As New DataDynamics.ActiveReports.DataSources.XMLDataSource()
xds.RecordsetPattern = "//prj/"
xds.LoadXML(xmlsource)
rpt.DataSource = xds
rpt.Run()
Viewer1.Document = rpt.Document
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
System.Xml.XmlDocument xdoc = new System.Xml.XmlDocument();
string xmlsource = "<prj>" +
    "<bcData>0003456</bcData>" +
    "<bcData>0003457</bcData>" +
    "<bcData>0003458</bcData>" +
    "<bcData>0003459</bcData>" +
    "</prj>";

rptEmpty rpt = new rptEmpty();
DataDynamics.ActiveReports.DataSources.XMLDataSource xds = new
DataDynamics.ActiveReports.DataSources.XMLDataSource();
xds.RecordsetPattern = "//prj/";
xds.LoadXML(xmlsource);
```

```

rpt.DataSource = xds;
rpt.FetchData += new
DataDynamics.ActiveReports.ActiveReport.FetchEventHandler(rpt_FetchData);
rpt.Run();
viewer1.Document = rpt.Document;

```

- Paste code like the following below the Form Load event to create a FetchData event for the report, access the XML data source, and pass in the string DataType attribute:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste BELOW the Form Load event.

```

Private Sub rpt_FetchData(ByVal sender As Object, ByVal eArgs As
DataDynamics.ActiveReports.ActiveReport.FetchEventArgs) Handles rpt.FetchData
    Try
        Dim r As DataDynamics.ActiveReports.ActiveReport
        Dim d As DataDynamics.ActiveReports.DataSources.XMLDataSource

        r = CType(sender, DataDynamics.ActiveReports.ActiveReport)
        d = CType(r.DataSource,
DataDynamics.ActiveReports.DataSources.XMLDataSource)

        Dim attr As System.Xml.XmlAttribute =
d.Document.CreateAttribute("DataType")
        attr.Value = "string"
        d.NodeList(d.CurrentPosition).Attributes.Append(attr)
    Catch
        eArgs.EOF = True
    End Try
End Sub

```


To write the code in C#

C# code. Paste BELOW the Form Load event.

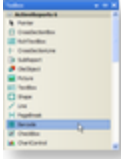
```

void rpt_FetchData(object sender,
DataDynamics.ActiveReports.ActiveReport.FetchEventArgs eArgs)
{
    try
    {
        DataDynamics.ActiveReports.ActiveReport r = sender as
DataDynamics.ActiveReports.ActiveReport;
        DataDynamics.ActiveReports.DataSources.XMLDataSource d = r.DataSource as
DataDynamics.ActiveReports.DataSources.XMLDataSource;
        System.Xml.XmlAttribute attr = d.Document.CreateAttribute("DataType");
        attr.Value = "string";
        d.NodeList[d.CurrentPosition].Attributes.Append(attr);
    }
    catch
    {
        eArgs.EOF = true;
    }
}

```

 **Note:** When you run this code, it transforms each `<bcData>0003456</bcData>` node into `<bcData DataType="string">0003456</bcData>`.

- Open the design view of NewActiveReport1 and from the ActiveReports 6 section of the Toolbox, drag a Barcode control and drop it onto the report.



- Barcode1 is selected in the Properties grid by default. In the **DataField** property enter . (a period) to assign the node to the barcode.



Add Hyperlinks

Using the Hyperlink property available on the following ActiveReports controls, you can add hyperlinks that connect to a Web page, open an e-mail, or jump to a bookmark.

- Label
- TextBox
- Picture
- OleObject



Note: When using the **HtmlViewer**, the **RawHtml**, the **Flash Viewer** or the **Silverlight Viewer** to display a report, you should indicate a full URL address (for example, "http://www.datadynamics.com") for the **Hyperlink** property.

To link to a Web page

- Click the control to select it.
- In the Properties window, set the **HyperLink** property to any valid URL.

To link to an e-mail address

- Click the control to select it.
- In the Properties window, set the **HyperLink** property to **mailto:** and any valid e-mail address.

To parse the URL out of a database field for a hyperlink

- Double-click the section where the control is located. This creates an event-handling method for the section's Format event.
- Add code to the Format event to
 - Parse the URL out of the HomePage field
 - Assign it to the HyperLink property of txtHomePage
 - Remove the URL markers from the text displayed in txtHomePage

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
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
```

To write the code in C#**C# code. Paste INSIDE the Format event.**

```
int iStart;
string sHTML;
if (txtHomePage.Text != "")
{
    iStart = txtHomePage.Text.IndexOf("#",0);
    sHTML = txtHomePage.Text.Substring(iStart, txtHomePage.Text.Length - iStart);
    sHTML = sHTML.Replace("#", "");
    txtHomePage.HyperLink = sHTML;
    txtHomePage.Text = txtHomePage.Text.Replace("#", "");
}
```

To create a hyperlink that jumps to a bookmark

1. Double-click the section where the control is located. This creates an event-handling method for the section's Format event.
2. Add code to the Format event to
 - Parse the URL out of the HomePage field
 - Assign it to the HyperLink property of txtHomePage
 - Remove the URL markers from the text displayed in txtHomePage

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

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste JUST ABOVE the Format event.**

```
Public pBM As New BookmarksCollection()
Dim iEntry As Integer
```

Visual Basic.NET code. Paste INSIDE the Format event.

```
Me.Detail1.AddBookmark(Me.txtCompanyName.Text)
Me.txtEntry.HyperLink = "toc://" + pBM(iEntry - 1).Label
Me.txtEntry.Text = pBM(iEntry - 1).Label
Me.txtPage.Text = pBM(iEntry - 1).PageNumber
```

To write the code in C#**C# code. Paste JUST ABOVE the Format event.**

```
public BookmarksCollection pBM = new BookmarksCollection();
int iEntry;
```

C# code. Paste INSIDE the Format event.

```
this.detail.AddBookmark(this.txtCompanyName.Text);
this.txtEntry.HyperLink = "toc://" + pBM[iEntry - 1].Label;
this.txtEntry.Text = pBM[iEntry - 1].Label;
this.txtPage.Text = pBM[iEntry - 1].PageNumber.ToString();
```

To display the page number of the bookmark in the table of contents

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(YourReportName Events)**.
2. At the top right of the code window, click the drop-down arrow and select **FetchData**. This creates an event-handling method for the report's FetchData event.
3. 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.NET code. Paste INSIDE the FetchData event.

```
If iEntry > pBM.Count - 1 Then
    eArgs.EOF = True
Else
    eArgs.EOF = False
    iEntry += 1
End If
```

To write the code in C#

1. Back in design view, click in the gray area below the report to select it.
2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
4. 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.

C# code. Paste INSIDE the FetchData event.

```
if (iEntry > pBM.Count - 1)
{
    eArgs.EOF = true;
}
else
{
    eArgs.EOF = false;
    iEntry += 1;
}
```

Add Annotations

You or your users can add notes, special instructions, even images, directly to the ActiveReport, making team collaboration, feedback, and support an easier task. Annotations are added in two ways: via the viewer's toolbar, or in code.

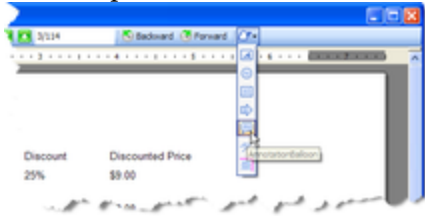
Annotations added via the viewer's toolbar are temporary. They reside on the Page object in which they are placed, and are destroyed when the report closes. In order to save annotations you must save the report data and accompanying annotations to RDF format.

Each annotation type allows you to change the colors, transparency, border, font, and alignment, plus other properties specific to the type of annotation. Available annotations include:

- **AnnotationText** A rectangular box in which you can enter text.
- **AnnotationCircle** A circle without text. You can change the shape to an oval.
- **AnnotationRectangle** A rectangular box without text.
- **AnnotationArrow** A 2D arrow in which you can enter text. You can change the arrow direction.
- **AnnotationBalloon** A balloon caption in which you can enter text. You can point the balloon's tail in any direction.
- **AnnotationLine** A line with text above or below it. You can add arrow caps to one or both ends and select different dash styles.
- **AnnotationImage** A rectangle with a background image and text. You can select an image and its position, and place text on the image.

To add annotations using the viewer

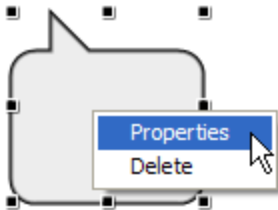
1. Load a report into the viewer and click the Annotations button on the toolbar.



2. Click the annotation you want to add and drag it onto the report.
3. Drag the corners to resize the annotation as needed, or drag the center to relocate it.

To change the properties of an annotation in the viewer

1. Right-click the annotation and select **Properties**.




2. In the Annotation Properties window that appears, add text, change the alignment, set colors, and use any other special properties to make the annotation appear the way you want it.

To save annotations

You can save annotations along with report data into an RDF file. The following example shows how to add a **Save Annotated Report** button to the viewer.

1. From the Visual Studio toolbox, drag a Button control onto the viewer.
2. Set the Text property of the button to **Save Annotated Report**.
3. Double-click the button. This creates an event-handling method for the button Click event.
4. Add code to the click handler to save the document to an RDF file.

 **Tip:** See **Save and Load Report Files (RDF)** for more information on loading the saved RDF file into the viewer.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the button Click event.**

```
Me.Viewer1.Document.Save("C:\\UserAnnotations.rdf")
```

To write the code in C#**C# code. Paste INSIDE the button Click event.**

```
this.viewer1.Document.Save("C:\\UserAnnotations.rdf");
```

To add annotations in code

The following example shows how to add annotations at run time and save the report data and annotations to an RDF file.

1. Double-click the title bar of the form in which you host the viewer. This creates an event-handling method for the form Load event.
2. Add code to the handler to run the report, add annotations, display the report in the viewer, and save it into an RDF file.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste ABOVE the class.**

```
Imports DataDynamics.ActiveReports.Document.Annotations
```

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New NewActiveReport1
'Run the report first
rpt.Run()

'Assign the viewer
Me.Viewer1.Document = rpt.Document

'Create an annotation and assign property values
Dim circle As New AnnotationCircle
circle.Color = System.Drawing.Color.GreenYellow
circle.Border.Color = System.Drawing.Color.Chartreuse

'Add the annotation
circle.Attach(1,1) 'screen location
Me.Viewer1.Document.Pages(0).Annotations.Add(circle)

'Set the size properties. The annotation must be added to the page first.
circle.Height = 0.25
circle.Width = 0.50

'Save annotations with the report in an RDF file
rpt.Document.Save("C:\\AnnotatedReport.rdf")
```

To write the code in C#**C# code. Paste ABOVE the class.**

```
using DataDynamics.ActiveReports.Document.Annotations;
```

C# code. Paste INSIDE the Form Load event.

```
NewActiveReport1 rpt = new NewActiveReport1();
//Run the report first
rpt.Run();

//Assign the viewer
this.viewer1.Document = rpt.Document;

//Create an annotation and assign property values
AnnotationCircle circle = new AnnotationCircle();
circle.Color = System.Drawing.Color.GreenYellow;
circle.Border.Color = System.Drawing.Color.Chartreuse;

//Add the annotation
circle.Attach(1,1); //screen location
this.viewer1.Document.Pages[0].Annotations.Add(circle);

//Set the size properties. The annotation must be added to the page first.
circle.Height = 0.25f;
circle.Width = 0.50f;

//Save annotations with the report in an RDF file
rpt.Document.Save("C:\\AnnotatedReport.rdf");
```

Export Reports

To export your reports to the various formats that ActiveReports supports, you must first add the export controls to your Visual Studio toolbox. For more information, see the **Adding ActiveReports Controls** topic. Here are the export formats that are included with ActiveReports:

- **HTML** For displaying in Web browsers or e-mail.
- **PDF** For preserving formatting on different computers.
- **RTF** For preserving some formatting, but allowing reports to be opened with Word or WordPad.
- **Text** For transmitting raw data, with little or no formatting.
- **TIFF** For transmitting faxes.
- **XLS** For spreadsheets.

To export a report

1. From the Visual Studio toolbox, drag the export filter that you want to use onto your Windows form. The control appears in the component tray below the form.
2. Double-click in the title bar of the form to create an event-handling method for the form Load event.
3. Add code to the event to run the report and export it.

The following examples show what the code for the method looks like for each of the export types.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New NewActiveReport1()
rpt.Run()
Me.HtmlExport1.Export(rpt.Document, Application.StartupPath + "\\HTMLExpt.html")
```



```
Me.PdfExport1.Export(rpt.Document, Application.StartupPath + "\\PDFExpt.pdf")
Me.RtfExport1.Export(rpt.Document, Application.StartupPath + "\\RTFExpt.rtf")
Me.TextExport1.Export(rpt.Document, Application.StartupPath + "\\TextExpt.txt")
Me.TiffExport1.Export(rpt.Document, Application.StartupPath + "\\TIFFExpt.tiff")
Me.XlsExport1.Export(rpt.Document, Application.StartupPath + "\\XLSExpt.xls")
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
NewActiveReport1 rpt = new NewActiveReport1();
rpt.Run();
this.htmlExport1.Export(rpt.Document, Application.StartupPath + "\\HTMLExpt.html");
this.pdfExport1.Export(rpt.Document, Application.StartupPath + "\\PDFExpt.pdf");
this.rtfExport1.Export(rpt.Document, Application.StartupPath + "\\RTFExpt.rtf");
this.textExport1.Export(rpt.Document, Application.StartupPath + "\\TextExpt.txt");
this.tiffExport1.Export(rpt.Document, Application.StartupPath + "\\TIFFExpt.tiff");
this.xlsExport1.Export(rpt.Document, Application.StartupPath + "\\XLSExpt.xls");
```

Create a Digital Signature for a PDF Export

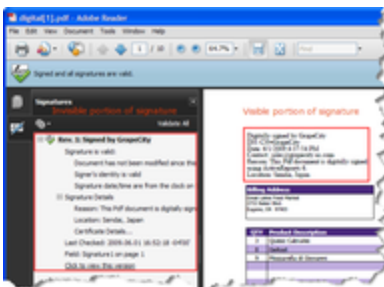
This feature is for use with the Professional Edition license only. An evaluation message is rendered when used with the Standard Edition license.

⚠ Important: In order to create a digital signature, you must first have a valid PKCS#12 certificate (*.pfx) file.

To use the code below, change the path and file name to point to your PFX, and change "password" to the password for your PFX file. You can use a third-party digital ID, or create a self-signed certificate.

For information on creating a self-signed certificate, see the [Adobe Acrobat Help topic "Create a self-signed digital ID."](#)

You can also create a PFX file from the Visual Studio command line. For more information and links to SDK downloads, see <http://www.source-code.biz/snippets/vbasic/3.htm>.



The following code samples assume that you have a report variable **rpt**, and that you have dragged the PDFExport object onto your form.

To add an invisible signature

The following example shows what the code for adding an invisible signature looks like. Replace the path and filename and password of the Certificate with your certificate information.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Me.PdfExport1.Signature.VisibilityType =
```

```

DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.Invisible
' Set certificate & password.
Me.PdfExport1.Signature.Certificate = New
Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath &
"\..\..\certificate.pfx", "password")

' Signature items.
Me.PdfExport1.Signature.Reason = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)("I agree.")
Me.PdfExport1.Signature.Location = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)("Japan")

Me.PdfExport1.Export(rpt.Document, Application.StartupPath &
"\..\..\VisibilityType_Invisible.pdf")

```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```

String path = Path.Combine(Application.StartupPath, @"\..\..\certificate.pfx");

String output = "output.pdf";
// Set certificate & password.
this.pdfExport1.Signature.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2(path, "password");


// Signature items.
this.pdfExport1.Signature.Reason = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField <String>("I agree.");
this.pdfExport1.Signature.Location = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField <String>("Japan");

this.pdfExport1.Export(rpt.Document, output);

```

To add a visible signature with the time stamp

The following example shows what the code for adding a visible signature with the time stamp looks like. Replace the path and filename and password of the Certificate with your certificate information, and replace the time stamping authority (TSA) URL with that of your TSA.

 **Note:** You must purchase a volume-based registration at an actual TSA for your applications.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```

' Text signature.
Me.PdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.Text
Me.PdfExport1.Signature.Stamp.Bounds = New RectangleF(1, 1, 4, 2)
Me.PdfExport1.Signature.Stamp.TextAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left

' Set certificate & password.
Me.PdfExport1.Signature.Certificate = New
Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath &
"\..\..\certificate.pfx", "password")

```

```

' Signature items.
Me.PdfExport1.Signature.SignDate = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of Date)
(System.DateTime.Now(), True)
Me.PdfExport1.Signature.Contact = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)
("ar6@grapecity.com", True)
Me.PdfExport1.Signature.Reason = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String) ("I agree.",
True)
Me.PdfExport1.Signature.Location = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String) ("Japan", True)

' Time stamp.
Me.PdfExport1.Signature.TimeStamp = New
DataDynamics.ActiveReports.Export.Pdf.Signing.TimeStamp("http://TSAServer", "null",
>null")
Me.PdfExport1.Export(rpt.Document, Application.StartupPath & "..\..\TimeStamped.pdf")

```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```

// Text signature.
this.pdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.Text;
this.pdfExport1.Signature.Stamp.Bounds = new RectangleF(1, 2, 5, 2);
this.pdfExport1.Signature.Stamp.TextAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left;

// Set certificate & password.
this.pdfExport1.Signature.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath
& "..\..\certificate.pfx", "password");

// Signature items.
this.pdfExport1.Signature.SignDate = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<System.DateTime>
(System.DateTime.Now, true);
this.pdfExport1.Signature.Contact = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<String>
("ar6@grapecity.com", true);
this.pdfExport1.Signature.Reason = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<String>("I agree.", true);
this.pdfExport1.Signature.Location = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<String>("Japan", true);

// Time stamp.
this.pdfExport1.Signature.TimeStamp = new
DataDynamics.ActiveReports.Export.Pdf.Signing.TimeStamp("http://TSAServer", "null",
>null);

this.pdfExport1.Export(rpt.Document, Application.StartupPath &
"..\..\TimeStamped.pdf");

```

To add a visible signature with text and graphics elements

The following example shows what the code for adding a visible signature with text and graphics looks like. Replace the path and filename and password of the Certificate with your certificate information.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
' ImageText signature.
Me.PdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.ImageText

' Bounds (Container of Text & Image).
Me.PdfExport1.Signature.Stamp.Bounds = New RectangleF(2, 1, 5, 1)

' Text area.
Me.PdfExport1.Signature.Stamp.TextAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left
Me.PdfExport1.Signature.Stamp.Font = New
Font(System.Drawing.FontFamily.GenericSansSerif, 8, FontStyle.Regular)
' Note: Specify (x, y) in relative coordinate from Bounds top-left.
Me.PdfExport1.Signature.Stamp.TextRectangle = New RectangleF(2, 0, 3, 1)

' Image area.
Me.PdfExport1.Signature.Stamp.Image = Image.FromFile(Application.StartupPath &
"\..\..\image\stamp.bmp")
Me.PdfExport1.Signature.Stamp.ImageAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Center
' Note: Specify (x, y) in relative coordinate from Bounds top-left.
Me.PdfExport1.Signature.Stamp.ImageRectangle = New RectangleF(0, 0, 2, 1)

' Set certificate & password.
Me.PdfExport1.Signature.Certificate = New
Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath &
"\..\..\certificate.pfx", "password")

' Signature items.
Me.PdfExport1.Signature.SignDate = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of Date)
(System.DateTime.Now(), True)
Me.PdfExport1.Signature.Contact = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)
("ar6@grapecity.com", True)
Me.PdfExport1.Signature.Reason = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)("I agree.",
True)
Me.PdfExport1.Signature.Location = New
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField(Of String)("Japan", True)

Me.PdfExport1.Export(rpt.Document, Application.StartupPath &
"\..\..\TextAndGraphics.pdf")
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
// ImageText signature.
this.pdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.ImageText;
```

```

// Bounds (Container of Text & Image).
this.pdfExport1.Signature.Stamp.Bounds = new RectangleF(2, 1, 5, 1);

// Text area.
this.pdfExport1.Signature.Stamp.TextAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left;
this.pdfExport1.Signature.Stamp.Font = new Font("Comic Sans MS", 8, FontStyle.Regular);
// Note: Specify (x, y) in relative coordinate from Bounds top-left.
this.pdfExport1.Signature.Stamp.TextRectangle = new RectangleF(2, 0, 3, 1);

// Image area.
this.pdfExport1.Signature.Stamp.Image = Image.FromFile("stamp.bmp");
this.pdfExport1.Signature.Stamp.ImageAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Center;
// Note: Specify (x, y) in relative coordinate from Bounds top-left.
this.pdfExport1.Signature.Stamp.ImageRectangle = new RectangleF(0, 0, 2, 1);

// Set certificate & password.
this.pdfExport1.Signature.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2("c:\\certificate.pfx",
"password");

//Signature items.
this.pdfExport1.Signature.SignDate = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<System.DateTime>
(System.DateTime.Now, true);
this.pdfExport1.Signature.Contact = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<string>
("ar6@grapecity.com", true);
this.pdfExport1.Signature.Reason = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<string>("I agree.", true);
this.pdfExport1.Signature.Location = new
DataDynamics.ActiveReports.Export.Pdf.Signing.SignatureField<string>("Japan", true);

this.pdfExport1.Export(rpt.Document, "c:\\TextAndGraphics.pdf");

```

To add a visible signature with graphics

The following example shows what the code for adding a visible signature with graphics looks like. Replace the path and filename and password of the Certificate with your certificate information.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```

' Image signature.
Me.PdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.Image
Me.PdfExport1.Signature.Stamp.Image = Image.FromFile(Application.StartupPath &
"..\\..\\image\\stamp.bmp")
Me.PdfExport1.Signature.Stamp.Bounds = New RectangleF(1, 2, 4, 1)
Me.PdfExport1.Signature.Stamp.ImageAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left

' Set certificate & password.
Me.PdfExport1.Signature.Certificate = New
Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath &

```

```
"\..\..\certificate.pfx", "password")

Me.PdfExport1.Export(rpt.Document, Application.StartupPath &
"\..\..\VisibilityType_Image.pdf")
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
// Image signature.
this.pdfExport1.Signature.VisibilityType =
DataDynamics.ActiveReports.Export.Pdf.Signing.VisibilityType.Image;
this.pdfExport1.Signature.Stamp.Image = Image.FromFile("c:\\stamp.bmp");
this.pdfExport1.Signature.Stamp.Bounds = new RectangleF(1, 2, 4, 1);
this.pdfExport1.Signature.Stamp.ImageAlignment =
DataDynamics.ActiveReports.Export.Pdf.Signing.Alignment.Left;

// Set certificate & password.
this.pdfExport1.Signature.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2("c:\\certificate.pfx",
"password");

this.pdfExport1.Export(rpt.Document, "c:\\VisibilityType_Image.pdf");
```

To add a certifying signature

The following example shows what the code for adding a certifying signature with form filling permission looks like. Replace the path and filename and password of the Certificate with your certificate information.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Me.PdfExport1.Signature.Certificate = New
System.Security.Cryptography.X509Certificates.X509Certificate2(Application.StartupPath
& "\..\..\certificate.pfx", "password")
Me.PdfExport1.Signature.CertificationLevel =
DataDynamics.ActiveReports.Export.Pdf.Signing.CertificationLevel.FormFilling
Me.PdfExport1.Export(rpt.Document, Application.StartupPath &
"\..\..\Certified_FormFilling.pdf")
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
this.pdfExport1.Signature.Certificate = new
System.Security.Cryptography.X509Certificates.X509Certificate2("c:\\certificate.pfx",
"password");
this.pdfExport1.Signature.CertificationLevel =
DataDynamics.ActiveReports.Export.Pdf.Signing.CertificationLevel.FormFilling;
this.pdfExport1.Export(rpt.Document, "c:\\Certified_FormFilling.pdf");
```

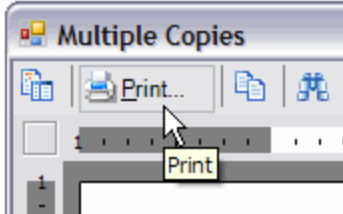
Print Multiple Copies, Duplex, and Landscape

With ActiveReports, printer settings can be modified at design time and at run time.

Multiple Copies

To set multiple copies in the print dialog

1. With a report displayed in the viewer, click **Print**.



2. In the Print dialog that appears, next to **Number of copies**, select the number of copies that you want to print.

To use code to set multiple copies

1. Double-click in the gray section below the report to create an event-handling method for the report's ReportStart event.
2. 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.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the ReportStart event.

```
Me.Document.Printer.PrinterSettings.Copies = 5
```

Visual Basic.NET code. Paste **INSIDE** the ReportEnd event.

```
Me.Document.Print(false, false)
```

To write the code in C#

C# code. Paste **INSIDE** the ReportStart event.

```
this.Document.Printer.PrinterSettings.Copies = 5;
```

C# code. Paste **INSIDE** the ReportEnd event.

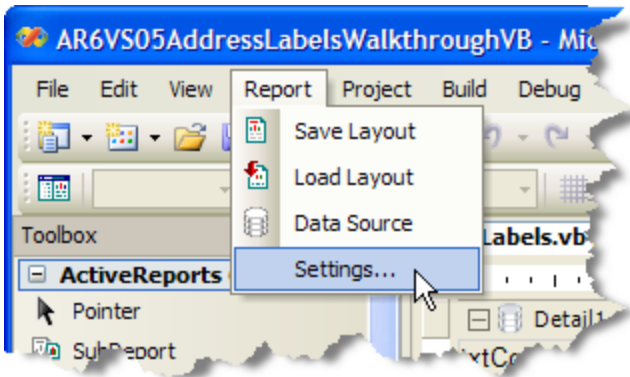
```
this.Document.Print(false, false);
```

Printer Settings

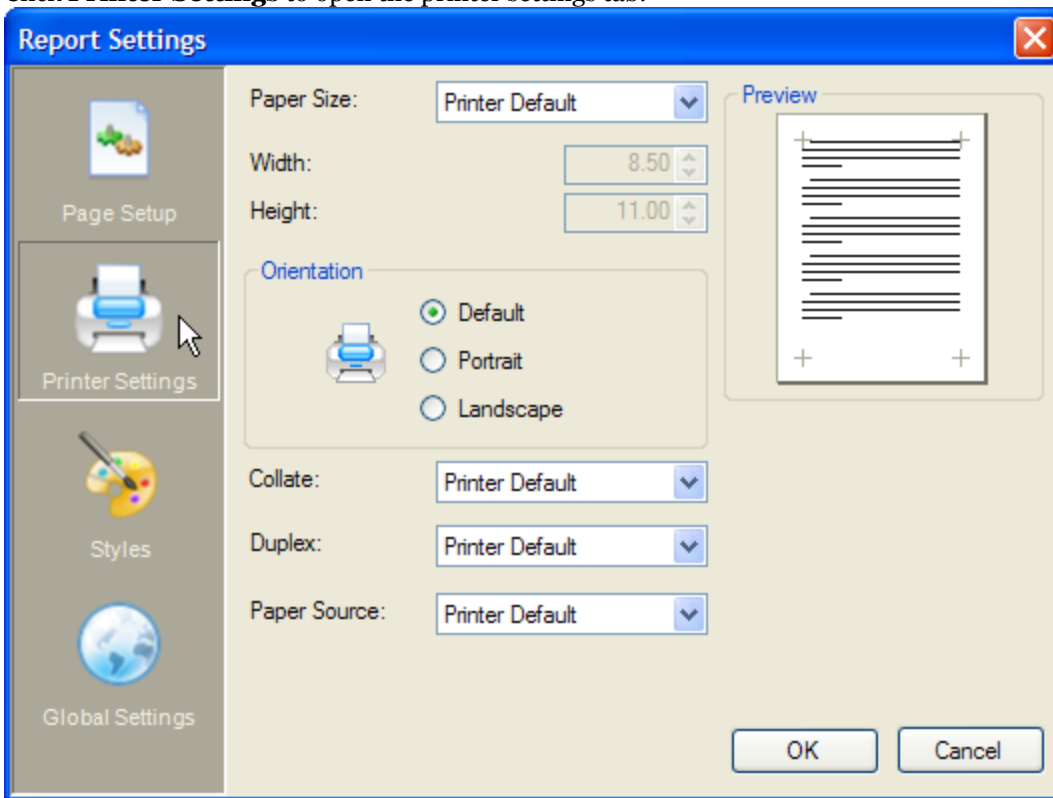
At design time, you can set paper size, orientation, collation, duplexing or paper source on the **Printer Settings** tab of the Report Settings window.

To open the Report Settings window

1. Open an ActiveReport.
2. Click on any section of the report to select it so that the Report menu appears.
3. Drop down the **Report** menu and select **Settings**.



4. Click **Printer Settings** to open the printer settings tab.



Duplex

To set duplexing in Printer Settings

1. Open the Report Settings window and select **Printer Settings**.
2. Next to **Duplex**, select one of the following options:
 - **Printer Default**: The report uses 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.
3. Click **OK** to return to the report.

Orientation

To change page orientation in Printer Settings

1. Open the Report Settings window and select **Printer Settings**.
2. In the Orientation section, select either **Portrait** or **Landscape**.
3. Click **OK** to return to the report.

To use code to change page orientation

1. Double-click in the gray section below the report to create an event-handling method for the report's ReportStart event.
2. Add code to the handler to change the page orientation of the report for printing.



Note: Page orientation can only be modified before the report runs. Otherwise, changes made to the page orientation are not used during printing.

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

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the ReportStart event.**

```
Me.PageSettings.Orientation = PageOrientation.Landscape
```

To write the code in C#**C# code. Paste INSIDE the ReportStart event.**

```
this.PageSettings.Orientation =  
DataDynamics.ActiveReports.Document.PageOrientation.Landscape;
```

Conditionally Show or Hide Details

You can use conditions in the Format event to control when the report's Detail section is shown.

| Product ID | Product Name | Units In Stock | Reorder Level |
|------------|----------------------------------|----------------|-----------------|
| 60 | Camembert Pierrot | 19 | Need to Reorder |
| 18 | Camraron 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 |

To add code to hide the detail section

1. Double-click the detail section of the report to create an event-handling method for the Detail Format event.
2. Add code to the handler to hide the detail section if the product is discontinued.

The following example shows what the code for the method looks like. This code assumes that your report has the following fields:

- txtReorderLevel (numeric)
- txtDiscontinued (Boolean)

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
If Me.txtReorderLevel.Value = 0 And Me.txtDiscontinued.Value = False Then
    Me.Detail1.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
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```

if(txtReorderLevel.Value == 0 && txtDiscontinued.Text == 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;
}

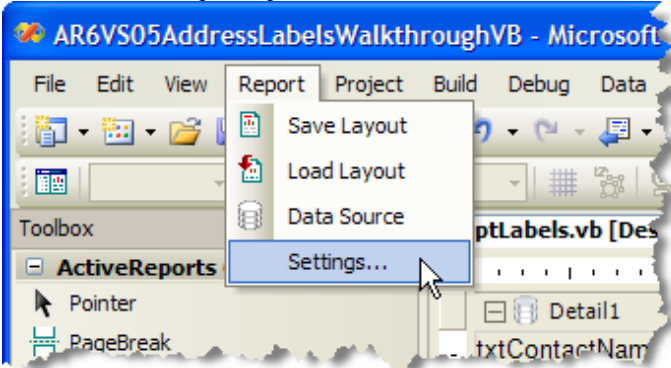
```

Use External Style Sheets

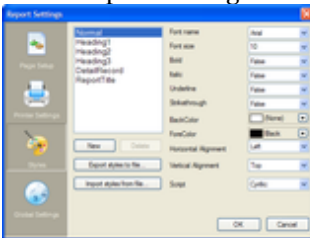
You can set custom style values using the Report Settings window, and then apply the styles to controls using the `ClassName` property in the Properties Window. If you want to apply these same styles to controls in other reports without setting them up each time, you can export them to XML files of type *.reportstyle which you can then select in the Report Settings window. Reports using external style sheets have these styles applied before the report is run.

To save an external style sheet

1. With an ActiveReport open and selected in Visual Studio, drop down the **Report** menu and select **Settings**.



2. In the Report Settings window that appears, click the **Styles** button to view the style settings.



3. By default, ActiveReports has six predefined styles: Normal, Heading1, Heading2, Heading3, DetailRecord, and ReportTitle. Click each of these styles in the list to modify them using the fields to the right, or click the **New** button to create a new style.
4. To save your styles to an external XML *.reportstyle file, click the **Export styles to file** button.
5. In the Save As dialog that appears, navigate to the location in which you want to save the style sheet, provide a name for the file, and click the **Save** button.
6. Back on the Report Settings window, click the **OK** button to close the window and save the styles in the current report.

To import an external style sheet at design time

1. With an ActiveReport open and selected in Visual Studio, drop down the **Report** menu and select **Settings**.
2. In the Report Settings window that appears, click the **Styles** icon to view the style settings.
3. Click the **Import styles from file** button. A message box warns that current styles will be deleted. Click **Yes** to continue.
4. In the Open dialog that appears, navigate to the *.reportstyle file that you want to use and click the **Open** button.

To apply an external style sheet at run time

To make a style sheet available at run time, double-click the grey area of the report to create an event-handling method for the ReportStart event of the report. Add code inside the handler to make the style style sheet available to the report.

The following examples show what the code for the method looks like.

Visual Basic.NET code. Paste **INSIDE** the ReportStart event.

```
Me.LoadStyles("C:\My.reportstyle")
```

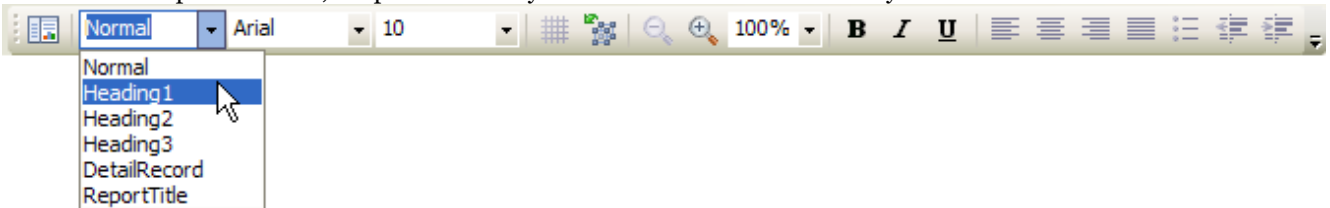
C# code. Paste **INSIDE** the ReportStart event.

```
this.LoadStyles("C:\My.reportstyle");
```

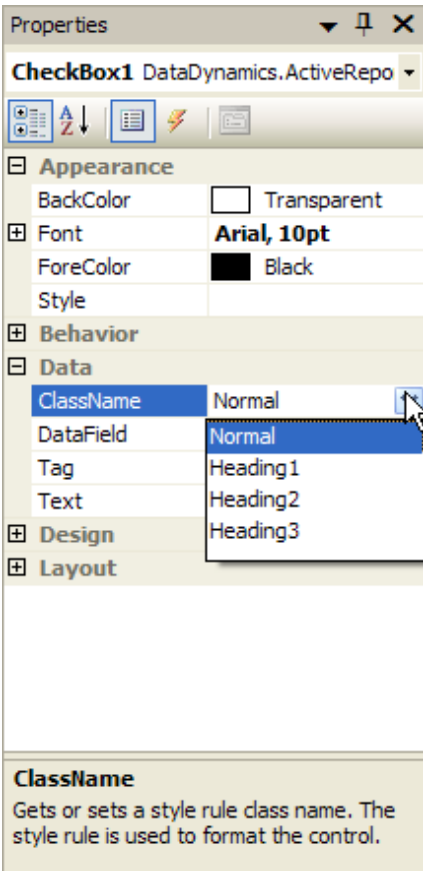
To apply styles to controls at design time

You can apply styles to four types of ActiveReports controls: **CheckBox**, **Label**, **TextBox**, and **ReportInfo**.

1. Select the control to which you want to apply the style.
2. In the ActiveReports toolbar, drop down the Styles combo box and select the style.



3. Or in the Properties Window, drop down the **ClassName** field and select the style to apply.



- When you run the report, ActiveReports applies the default style values for the selected style, or the style values contained in the specified external style sheet.

To apply styles to controls at run time

You can apply styles to four types of ActiveReports controls: **CheckBox**, **Label**, **TextBox**, and **ReportInfo**.

To apply a style at run time, double-click the section of the report containing the control to create an event-handling method for the Format event of the section. Add code inside the handler to apply the style to the control.

The following examples show what the code for the method looks like.

Visual Basic.NET code. Paste INSIDE the Format event.

```
Me.TextBox1.ClassName = "Heading1"
```

C# code. Paste INSIDE the Format event.

```
this.textBox1.ClassName = "Heading1";
```

Add Bookmarks

ActiveReports can display bookmarks and nested bookmarks in the viewer's table of contents for fields, groups, subreports. You can also add special bookmarks at run time.

To set up basic bookmarks

This code uses the same controls used in the **Basic Data Bound Reports** walkthrough

- Double-click on the detail section of the report. This creates an event-handling method for the report's Detail Format

event.

2. Add code to the handler to set up bookmarks.

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
Me.Detail1.AddBookmark(txtProductName1.text)
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```
detail.AddBookmark(txtProductName1.Text);
```

To set up leveled or nested bookmarks

This code uses fields from the **Group On Unbound Fields** walkthrough.

1. Double-click on the detail section of the report. This creates an event-handling method for the report's Detail Format event.
2. Add code to the handler to set up bookmarks.

The following example shows what the code to set up leveled or nested Bookmarks looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
Me.Detail1.AddBookmark(txtCategoryName.Text + "\" + txtProductName.Text)
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```
detail.AddBookmark(txtCategoryName.Text + "\\\" + txtProductName.Text);
```

To nest grandchild bookmarks and use bookmarks in grouping

1. Double-click in the Detail section of the report. This creates an event-handling method for the report's Detail_Format event.
2. Add code to the handler to set up a bookmark for each city and nest city bookmarks within each country, and company bookmarks in each city.

The following example shows what the code for the detail section looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
Me.Detail1.AddBookmark(txtCountry1.Text + "\" + txtCity1.Text + "\" + txtCompanyName1.Text)
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```
this.detail.AddBookmark(txtCountry1.Text + "\\\" + txtCity1.Text + "\\\" +  
txtCompanyName1.Text);
```

1. Double-click in the Group Header section of the report. This creates an event-handling method for the report's Group Header Format event.
2. Add code to the handler to set up a bookmark for each instance of the country group.

The following example shows what the code for the group header looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Group Header Format event.

```
Me.GroupHeader1.AddBookmark(txtCountry1.Text)
```

To write the code in C#

C# code. Paste INSIDE the Group Header Format event.

```
this.groupHeader1.AddBookmark(txtCountry1.Text);
```

To combine parent report and subreport bookmarks

This code uses the same controls as those found in the **Subreports with Run-Time Data Sources** walkthrough.

1. Double-click in the Detail section of the **main** report to create an event-handling method for the report's Detail Format event.
2. Add code to the handler to create a bookmark for each instance of the CategoryName field in the main report.

The following example shows what the code for the method looks like for the main report.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event of the main report.

```
Me.Detail1.AddBookmark(txtCategoryName1.Text)
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event of the main report.

```
detail1.AddBookmark(txtEmployeeID1.Text);
```

1. Double-click in the Detail section of the **subreport** to create an event-handling method for the report's Detail Format event.
2. Add code to the handler to create a bookmark for each instance of the CategoryName field in the subreport.

The following example shows what the code for the method looks like for the subreport.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event of the subreport.

```
Me.Detail1.AddBookmark(CType(Me.ParentReport.Sections("Detail1").Controls("txtCategoryName1"),  
TextBox).Text + "\" + Me.txtProductName.Text)
```

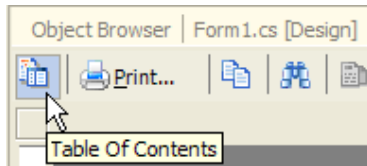
To write the code in C#

C# code. Paste INSIDE the Detail Format event of the subreport.

```
this.detail1.AddBookmark(((TextBox)  
(this.ParentReport.Sections["ghEmployees"].Controls["txtEmployeeID1"])).Text + "\" +  
this.txtCompanyName1.Text);
```

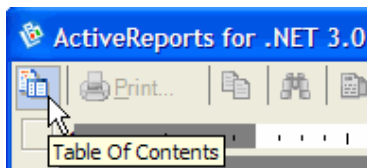
To preview the report and Bookmarks Collection in the designer

1. Click the **Preview** tab at the bottom of the designer.
2. Click the **Table of Contents** icon to view the Bookmarks collection.



To view a report's bookmarks in the viewer

1. Add the ActiveReports viewer control to your Windows form.
2. Add code to display the report document in the viewer. See **Viewing Reports** for help.
3. Press **F5** to run the report.
4. Click the **Table of Contents** icon to view the Bookmarks collection.



To add special bookmarks at run time

To create and add special bookmarks to the bookmarks collection at run time, add the bookmarks to the report document's pages collection.

Caution: Keep in mind that the pages collection does not exist until after the report runs, so use this code in the ReportEnd event or in form code after the report has run.

To write the code in Visual Basic.NET

1. In design view of the report, drop down the field at the top left of the code view and select **(rptYourReportName Events)**.
2. Drop down the field at the top right of the code view and select **ReportEnd**. This creates an event-handling method for ReportEnd event.
3. Add code to the handler to add a bookmark.

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

Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
Me.Document.Pages(0).AddBookmark("New Bookmark", 1)
```

To write the code in C#

1. Click in the gray area below the report to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportEnd**. This creates an event-handling method for the ReportEnd event.
4. Add code to the handler to add a bookmark.

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

C# code. Paste INSIDE the ReportEnd event.

```
this.Document.Pages[0].AddBookmark("New Bookmark", 1);
```

Insert or Add Pages

ActiveReports allows you to run multiple reports and combine their PagesCollections, or specified portions of them, into

a single report. The document containing the merged reports can be saved to an RDF file or exported.

To add pages from one report to another

The Add method of the ActiveReports Document Pages Collection takes one parameter: **value**. The **value** parameter references a report document page. To add an entire report, use code like that in the example below to iterate through the entire pages collection of a report and append it to the first report.

1. Add the ActiveReports viewer control to the Windows Form. For more information, see the **Adding ActiveReports Controls** topic.
2. Double-click the title bar of the Windows Form to create an event-handling method for the form's Load event.
3. Add code to the handler to add rptTwo to rptOne

The following example shows what the code for the Add() method looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste JUST ABOVE the Form Load event.

```
Dim i As Integer
```

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
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
```

To write the code in C#

C# code. Paste JUST ABOVE the Form Load event.

```
int i;
```

C# code. Paste INSIDE the Form Load event.

```
rptOne rpt = new rptOne();  
rpt.Run();  
rptTwo rpt2 = new rptTwo();  
rpt2.Run();  
for(i = 0; i < rpt2.Document.Pages.Count; i++)  
{  
    rpt.Document.Pages.Add(rpt2.Document.Pages[i]);  
}  
viewer1.Document = rpt.Document;
```

To add a range of pages from one report to another

The AddRange method has two overloads, each with one parameter. The first overload takes an array of page objects, while the second takes a pages collection. Use the second overload to add an entire report's pages collection. Use the first (as in the example below) to append only specified pages from the second report onto the first.

1. Add the ActiveReports viewer control to the Windows Form.
2. Double-click the title bar of the Windows Form to create an event-handling method for the form's Load event.
3. Add code to the handler to use the AddRange() method to add rptTwo to rptOne

The following example shows what the code for the AddRange() method looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New rptOne()  
rpt.Run()  
Dim rpt2 As New rptTwo()  
rpt2.Run()  
rpt.Document.Pages.AddRange(New Page()  
{rpt2.Document.Pages(0), rpt2.Document.Pages(1)})  
Viewer1.Document = rpt.Document
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptOne rpt = new rptOne();  
rpt.Run();  
rptTwo rpt2 = new rptTwo();  
rpt2.Run();  
rpt.Document.Pages.AddRange(new Page[]  
{rpt2.Document.Pages[0], rpt2.Document.Pages[1]} );  
viewer1.Document = rpt.Document;
```

To insert pages from one report into another

The Insert method takes two parameters, an **index**, which dictates where in the main report the pages are inserted, and **value**, the report page to insert.

1. Add the ActiveReports viewer control to the Windows Form.
2. Double-click the title bar of the Windows Form to create an event-handling method for the form's Load event.
3. Add code to the handler to insert a second report at the beginning of the first

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New rptInsertPage()  
rpt.Run()  
Dim rpt2 As New rptCoverPage()  
rpt2.Run()  
rpt.Document.Pages.Insert(1, rpt2.Document.Pages(0))  
Viewer1.Document = rpt.Document
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptInsertPage rpt = new rptInsertPage();  
rpt.Run();  
rptCoverPage rpt2 = new rptCoverPage();  
rpt2.Run();  
rpt.Document.Pages.Insert(1, rpt2.Document.Pages[0]);  
viewer1.Document = rpt.Document;
```

To add code to insert a page into a specified report location

The InsertNew method takes one parameter, **index**, which specifies the page after which you want to insert a new blank page.

1. Double-click the title bar of the Windows Form to create an event-handling method for the form's Load event.
2. Add code to the handler to insert a page into a specific report location.

The following example shows what the code for the InsertNew() method looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New rptInsertPage()
rpt.Run()
rpt.Document.Pages.InsertNew(3)
Viewer1.Document = rpt.Document
```


To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptInsertPage rpt = new rptInsertPage();
rpt.Run();
rpt.Document.Pages.InsertNew(3);
viewer1.Document = rpt.Document;
```

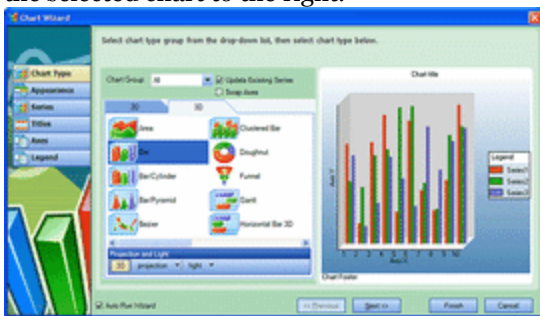
Create Charts

The ActiveReports ChartControl offers 45 chart types which, along with many other properties, allow you to create virtually any type of chart that you can conceive. The fastest way to create a chart is to use the Chart Wizard.

 **Note:** If your Chart Wizard does not appear when you add a ChartControl to a report, see **Access the Chart Wizard and Data Source** for information.

To create a chart using the Chart Wizard

1. From the ActiveReports toolbox, drag the ChartControl onto a report.
2. In the Chart Wizard that appears, the initial page displays the available 3D chart types, along with a preview of the selected chart to the right.

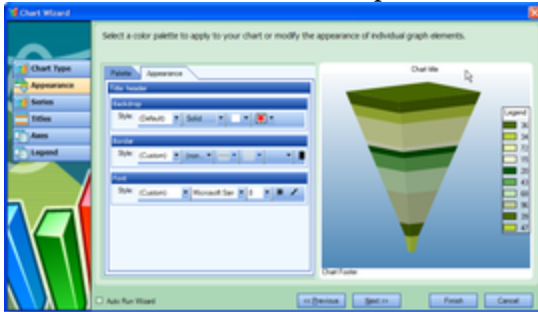


3. Select the type of chart that you want to create. You can scroll to the right to view more charts, or select the 2D tab. You can also drop down the Chart Group field to limit the chart types displayed to Area, Bar, Financial, Line, Pie/Doughnut, or Point/Bubble. Some of the chart types are available only as 2D charts.

4. Still on the Chart Type tab, you can further configure the chart by selecting the **Swap Axes** checkbox. If you are using a 3D chart, you can also change the **projection** and **light** settings. To do so, click the arrow to drop down the window for each item.
5. When you have the chart type configured the way you want it, click the **Next** button to move on to the Appearance page.

Appearance page

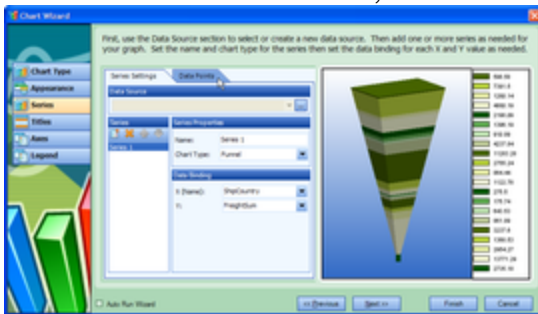
1. The Appearance page has two tabs: one allows you to select a color scheme, and the other allows you to select individual elements of the chart preview and select appearance settings for them.




2. Select a palette to set the color scheme, then click the Appearance tab.
3. Click the different areas of the preview chart, such as the title, footer, legend, legend title, backdrop, and the chart itself. Each reveals a different set of properties that you can use to control the appearance of the element.
4. When you finish with the appearance settings, click the **Next** button to move on to the Series page.

Series page

1. The Series page has two tabs: one allows you to set the data source for the chart and bind data fields to X and Y values to each series in the chart, and the other allows you to view the data values for X and Y.



2. Next to Data Source, click the ellipsis button to open the Chart Data Source dialog.
3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to your database or the sample Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter a SQL query to select the data that you want.

 **Tip:** A commonly used SQL Query for charts on the Nwind.mdb sample database is:

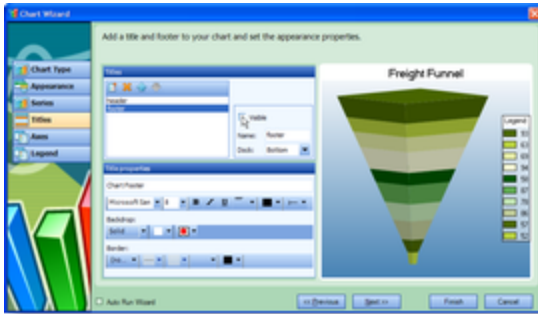
SQL Query. Paste in the Query textbox.

```
SELECT ShipCountry, SUM(Freight) AS FreightSum FROM Orders GROUP BY ShipCountry
```

8. Click **OK** to save the data source and return to the Chart Wizard.
9. Select **Series1** in the list of series, and set its name and chart type in the fields to the right.
10. Drop down the fields in the Data Binding section to select X and Y values. The X value takes a string field, while the Y value takes a numeric field.
11. If you do not need more than one series, delete **Series2** and **Series3**.
12. Click the Data Points tab to view the bound data or change the values.
13. When you finish setting up the data series for the chart, click the **Next** button to go to the Titles page.

Titles page

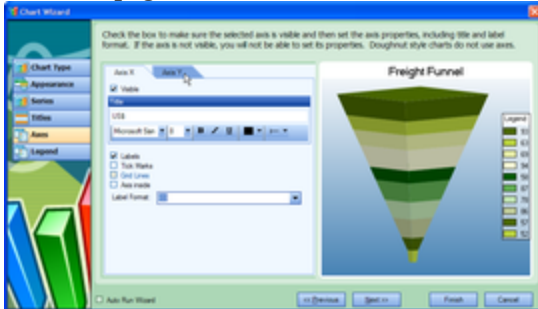
1. There is only one tab on the Titles page. Select the header or footer in the list, and change its properties in the fields below.



2. If you do not want to include a title, either delete it or clear the **Visible** checkbox.
3. Otherwise, set the title text and increase the font size, and click the **Next** button to go to the Axes page.

Axes page

1. The Axes page has two tabs: one for Axis X and one for Axis Y.



2. On the **Axis X** tab, enter a title for the axis, and set the font size and other font properties.
3. Select the check boxes next to any of the properties that you want to apply to the X axis.
4. Enter or select a label format.

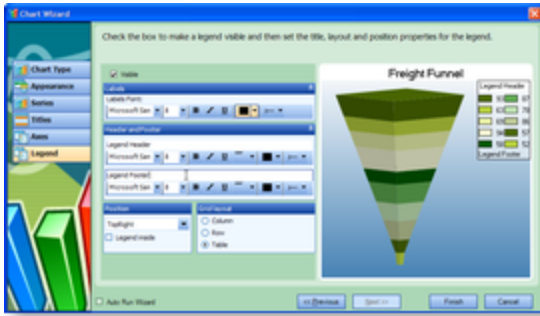


Tip: The ActiveReports chart control uses standard Visual Studio .NET formatting syntax. The format is {Tag : Format}. For example, {Value:C} formats the text as currency. {Value:D} formats the text as a date.

5. When you have set up Axis X, click the **Axis Y** tab to set its properties.
6. When you have finished, click the **Next** button to go to the Legend page.

Legend page

1. There is only one tab on the Legend page. Use it to set up the appearance of the legend.



2. If you do not want to display a legend for the chart, clear the **Visible** check box.
3. Otherwise, set font properties for the labels in the Labels section.
4. Enter text to display in the legend header and footer, and set font properties on the text in the Header and Footer section.
5. In the Position section, select the position relative to the chart in which to display the legend. You can also select the **Legend inside** check box to place the legend inside the chart.
6. In the Grid Layout section, select the layout for the legend items.
7. When you have finished, click the **Finish** button to save the changes and close the Chart Wizard.

Access the Chart Wizard and Data Source

You can open the chart wizard, chart data source, and other chart-related functions by clicking verbs in the Properties window.

To access verbs in the Properties window if they are not displayed by default

1. Right-click anywhere in the Properties window.
2. Select **Commands** so that it becomes checked.



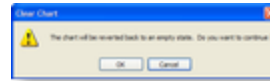
3. The verbs display at the bottom of the Properties window. You may need to resize the verb area in order to see all six of them.



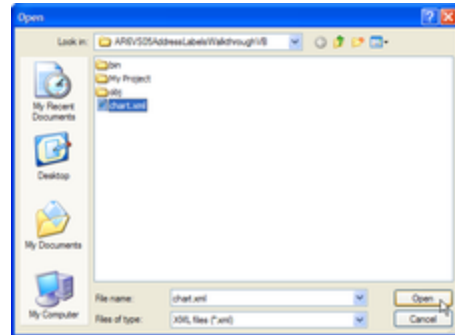
Here is a table describing each of the six verbs.

| Verb | Usage | Window |
|------|-------|--------|
|------|-------|--------|

Clear Chart Clears all data, including default data, from the chart. Click the **OK** button to clear the chart, or click **Cancel**.

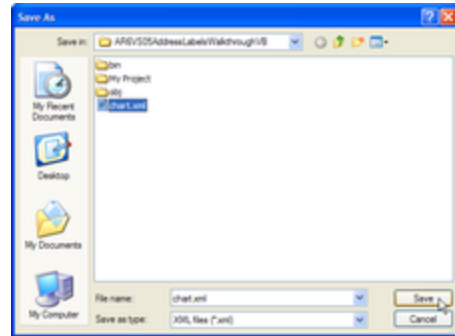


Load... Loads a chart previously saved to XML format. In the Open window that appears, navigate to the XML file, select it, and click the **Open** button to load the saved chart into the current chart control.

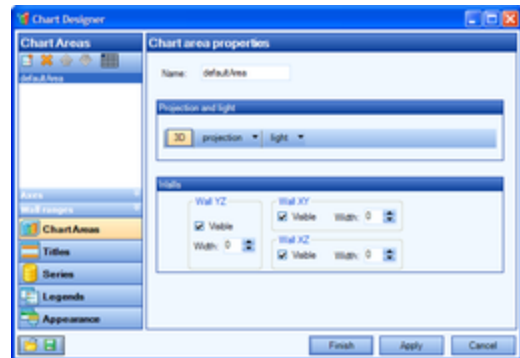


Save As... Saves all data from a chart into XML format. In the Save As window that appears, navigate to the directory in which you want to save the XML file, enter a File Name, and click the **Save** button to save the current chart's settings to XML format.

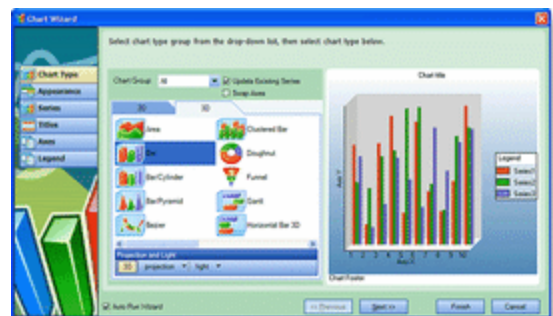
Saved charts can be loaded into chart controls on other reports.



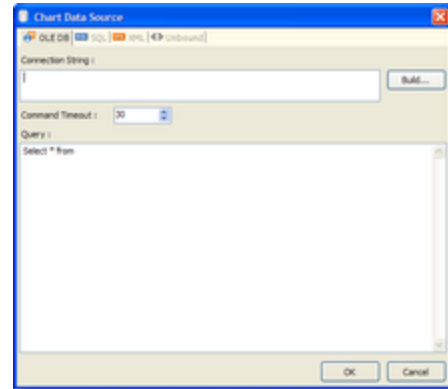
Customize... Opens the Chart Designer window, where you can change settings on the chart areas, titles, series, legends, and appearance of the chart.



Wizard... Opens the Chart Wizard, which takes you through the basic steps of creating a chart.



Data Source... Opens the Chart Data Source window, from which you can connect the chart to any data source.



Load a File into a RichText Control

You can load an RTF file or an HTML file into the ActiveReports RichText control at run time.

Caution: Do not attempt to load a file into a RichTextBox in a section that repeats. After the first iteration of the section, the RTF or HTML file is in use by that first iteration and returns "file in use" errors when the section is processed again.

To write an RTF file to load into a RichText control

1. Open WordPad, and paste the following formatted text and table into it.
Paste into an RTF File

Customer List by Country

Argentina

- Rancho grande
- Océano Atlántico Ltda.
- Cactus Comidas para llevar

Austria

- Piccolo und mehr
- Ernst Handel

Belgium

- Suprêmes délices
- Maison Dewey

Brazil

- Família Arquibaldo
- Wellington Improtadora
- Que Delícia
- Tradição Hipermercados

- Ricardo Adocicados
- Hanari Carnes
- Queen Cozinha
- Comércio Mineiro
- Gourmet Lanchonetes

| Month | Sales |
|----------|---------|
| October | \$4,872 |
| November | \$8,517 |
| December | \$9,623 |

2. Save the file as **sample.rtf** in the **debug** directory in the bin folder of your project.

To load the RTF file into the RichText control

1. Double-click the detail section of the report to create an event-handling method for the Detail Format event.
2. Add code to the handler to load the RTF file into the RichText control.



Note: The Application.Startup path code does not work in preview mode. **You must run the project in order to see the file load.**

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
Dim streamRTF As New System.IO.FileStream(System.Windows.Forms.Application.StartupPath
+ "\sample.rtf", System.IO.FileMode.Open)
Me.RichTextBox1.Load(streamRTF, RichTextType.Rtf)
```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```
System.IO.FileStream streamRTF = new
System.IO.FileStream(System.Windows.Forms.Application.StartupPath + "\\sample.rtf",
System.IO.FileMode.Open);
this.RichTextBox1.Load(streamRTF, RichTextType.Rtf);
```

To write a quick HTML file to load into a RichText control

1. Open Notepad and paste the following code into it.
HTML code

HTML code. Paste in a NotePad file.

```
<html>
<body>
<center><h1>Customer List by Country</h1></center>
<h1>Argentina</h1>
<ul>
<li>Rancho grande
<li>Océano Atlántico Ltda.
<li>Cactus Comidas para llevar
```

```


</ul>
<h1>Austria</h1>
<ul>
<li>Piccolo und mehr
<li>Ernst Handel
</ul>
<h1>Belgium</h1>
<ul>
<li>Suprêmes délices
<li>Maison Dewey
</ul>
<h1>Brazil</h1>
<ul>
<li>Familia Arquibaldo
<li>Wellington Improtadora
<li>Que Delícia
<li>Tradição Hipermercados
<li>Ricardo Adocicados
<li>Hanari Carnes
<li>Queen Cozinha
<li>Comércio Mineiro
<li>Gourmet Lanchonetes
</ul>
<table>
<tr><th>Month</th><th>Sales</th></tr>
<tr><td>October</td><td>$4,872</td></tr>
<tr><td>November</td><td>$8,517</td></tr>
<tr><td>December</td><td>$9,623</td></tr>
</table>
</body>
</html>

```

2. Save the file as **sample.html** in the **debug** directory in the bin folder of your project.

Loading the HTML file into a RichText control

1. Double-click the detail section of the report to create an event-handling method for the Detail Format event.
2. Add code to the handler to load the HTML file into the RichText control.

 **Note:** The Application.Startup path code does not work in preview mode. **You must run the project in order to see the file load.**

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```

Dim streamHTML As New System.IO.FileStream(System.Windows.Forms.Application.StartupPath
+ "\sample.HTML", System.IO.FileMode.Open)
Me.RichTextBox1.Load(streamHTML, RichTextType.Html)

```

To write the code in C#

C# code. Paste INSIDE the Detail Format event.

```


System.IO.FileStream streamHTML = new
System.IO.FileStream(System.Windows.Forms.Application.StartupPath + "\\sample.html",

```

```
System.IO.FileMode.Open);  
this.RextTextBox1.Load(streamHTML, RichTextType.Html);
```

Use Custom Controls on Reports (TreeView)

ActiveReports allows you to drop a third party control onto a report where it is recognized as a custom control. You can access its properties using type casting. In this case, we use hidden textboxes to populate a Visual Studio TreeView control.

 **Tip:** To access properties of a custom control, you should use type casting (for the samples of code, see **To access the TreeView control properties in code and assign data** section below).

To add the TreeView control to a report

1. From the Visual Studio toolbox Common Controls tab, drag the TreeView control onto the detail section of a report.
2. Notice that in the Properties window, the control is called CustomControl1.

To add data and hidden textboxes to the report

1. Connect the report to the sample nwind.mdb. (For help with this, see the **Bind Reports to a Data Source** topic.)
2. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Orders ORDER BY ShipCountry, ShipCity, CustomerID, EmployeeID
```

3. From the Report Explorer, drag the following fields onto the detail section of the report:
 - ShipCountry
 - ShipCity
 - CustomerID
 - EmployeeID
4. Select all four textboxes, and in the Properties window, change the **Visible** property to **False**.

To create a function to add nodes to the TreeView control

1. Right-click the report and select **View Code**.
2. Add a function to the report to add nodes to the TreeView.

The following examples show what the code for the function looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the report class.

```
Private Function AddNodeToTreeView(ByVal colNodes As TreeNodeCollection, ByVal sText As  
String) As TreeNode  
    Dim objTreeNode As TreeNode  
    objTreeNode = New TreeNode(sText)  
    colNodes.Add(objTreeNode)  
    Return objTreeNode  
End Function
```

To write the code in C#**C# code. Paste INSIDE the report class.**

```
private TreeNode AddNodeToTreeView(TreeNodeCollection colNodes, string sText)
{
    TreeNode objTreeNode;
    objTreeNode = new TreeNode(sText);
    colNodes.Add(objTreeNode);
    return objTreeNode;
}
```

To access the TreeView control properties in code and assign data

1. Back in the design view of the report, double-click the detail section to create an event-handling method for the Detail Format event.
2. Add code to the handler to access the TreeView properties and assign data from the hidden textboxes.

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

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Detail Format event.**

```
'type cast the custom control as a TreeView
ctlTreeView = CType(Me.CustomControl1.Control, TreeView)

'create a tree node
Dim objCountryTreeNode As TreeNode
'assign the text from a hidden textbox to the node
objCountryTreeNode = AddNodeToTreeView(ctlTreeView.Nodes, Me.txtShipCountry1.Text)
'add a second-level node
AddNodeToTreeView(objCountryTreeNode.Nodes, Me.txtShipCity1.Text)
'expand the top-level node so the second-level node is in view
objCountryTreeNode.Expand()

'create a second top-level node
Dim objCustomerTreeNode As TreeNode
objCustomerTreeNode = AddNodeToTreeView(ctlTreeView.Nodes, Me.txtCustomerID1.Text)
AddNodeToTreeView(objCustomerTreeNode.Nodes, Me.txtEmployeeID1.Text)
objCustomerTreeNode.Expand()
```

To write the code in C#**C# code. Paste INSIDE the Detail Format event.**

```
//type cast the custom control as a TreeView
ctlTreeView = (TreeView)this.CustomControl1.Control;

//create a tree node
TreeNode objCountryTreeNode;
//assign the text from a hidden textbox to the node
objCountryTreeNode = AddNodeToTreeView(ctlTreeView.Nodes, this.txtShipCountry1.Text);
//add a second-level node
AddNodeToTreeView(objCountryTreeNode.Nodes, this.txtShipCity1.Text);
//expand the top-level node so the second-level node is in view
objCountryTreeNode.Expand();
```

```
//create a second top-level node
TreeNode objCustomerTreeNode;
objCustomerTreeNode = AddNodeToTreeView(ctlTreeView.Nodes, this.txtCustomerID1.Text);
AddNodeToTreeView(objCustomerTreeNode.Nodes, this.txtEmployeeID1.Text);
objCustomerTreeNode.Expand();
```

Create Report Templates (Inheritance)

You can create a base report class as a template from which other reports can inherit. This is useful when many of your reports share common features, such as identical page headers and footers. Instead of recreating the look every time, create template headers and footers once and use inheritance to apply them to your other reports.

In ActiveReports, you can use inheritance at both design time and run time. A simple example of how you can use this functionality is a company letterhead template.


To create a company letterhead template at design time

1. Add an ActiveReport to your Visual Studio project and name the file **rptLetterhead**. This is the base report class.
2. Drag the following controls from the ActiveReports toolbox to the indicated section of rptLetterhead and set the properties as indicated. Reports do not inherit from the detail section, so do not add controls to it.

Controls for rptLetterhead

| Control | Section | Location | Size | Miscellaneous |
|---------|------------|---------------|--------------------|--|
| Picture | PageHeader | 0, 0 in | 3, 0.65 in | Image = (click ellipsis, navigate to <i>C:\Program Files\GrapeCity\ActiveReports 6\Introduction</i> (or to <i>C:\Program Files (x86)\GrapeCity\ActiveReports 6\Introduction</i> on a 64-bit Windows operating system) and select itopimage1.gif) PictureAlignment = TopLeft |
| Label | PageHeader | 0.75, 0.65 in | 1.2, 0.19 in | Text = Control Yourself! FontStyle = Bold |
| Label | PageFooter | 0, 0 in | 6.5, 0.19 in | Text = http://www.datadynamics.com HyperLink = http://www.datadynamics.com FontStyle = Bold Alignment = Center |

3. Click in the grey area below the report to select it, and in the Properties window, set the **MasterReport** property to **True**.

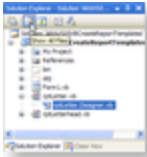
 **Important:** Do not set the MasterReport property to True until you have finished making changes to the report. Setting this property to True triggers major changes in the designer file for the report.



- The detail section of the report is disabled. When you create reports that inherit from this class, they will provide the detail section.



- Add a second ActiveReport to your project and name the file **rptLetter**. This report will inherit its PageHeader and PageFooter sections from rptLetterhead.
- In the Solution Explorer tool strip, click the **Show All Files** button.



- Expand the rptLetter node and double-click to open the **rptLetter.Designer** file (in a C# project, right-click the rptLetter.cs node and select **View Code** to display the code view for the report).
- In the rptLetter.Designer code (in a C# project, in the rptLetter code), change the inheritance statement so that rptLetter inherits from rptLetterhead instead of from DataDynamics.ActiveReports.ActiveReport. Use code like the following:

To write the code in Visual Basic.NET

Visual Basic.NET code.

```
Partial Public Class rptLetter
    Inherits YourProjectName.rptLetterhead
```

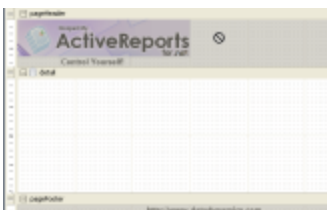
To write the code in C#

C# code.

```
public partial class rptLetter : YourProjectName.rptLetterhead
```

- Close the reports and from the Build menu, select **Rebuild**. When you reopen rptLetter, the inherited sections and controls are disabled.

Note: To make changes in these sections in rptLetterhead and rebuild your project again.



10. Add data and controls to the detail section of the report as you would any other report. See the **Basic Data Bound Reports** walkthrough for more information.

Caution: Base reports and the reports that inherit from them cannot contain controls with duplicate names. You can compile and run your project with duplicate control names, but you cannot save the layout until the duplicate names are changed.

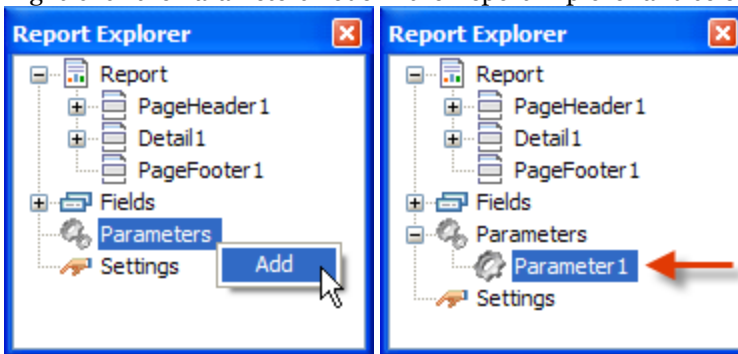
Add Parameters

There are several ways to use parameters in ActiveReports 6.

Report Explorer Parameters

To add parameters using the Report Explorer

1. Right-click the Parameters node in the Report Explorer and select **Add**. This adds a child to the Parameters node.



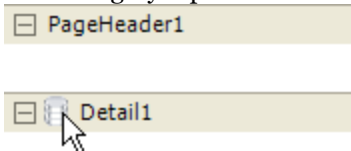
2. Once you add a parameter to the Report Explorer, you can set it up in the Properties window.
3. You can pass the parameter to a field on the report, or access it programmatically.

SQL Query Parameters

To add parameters to a SQL query

You can allow users to filter the amount of information exposed in a report through the use of parameters. When you add SQL parameters to a report, ActiveReports displays an Enter Report Parameters dialog where the user can enter the values to pull from the database.

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



2. On the **OLE DB** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to your database or the sample Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter a SQL query like the one below, which contains parameter syntax that ActiveReports uses to generate the following dialog.

SQL Query. Paste in the Report Data Source window's Query box.

```
SELECT * FROM Products
INNER JOIN (Orders INNER JOIN [Order Details] ON Orders.OrderID= [Order
Details].OrderID) ON Products.ProductID = [Order Details].ProductID WHERE
Products.SupplierID = <%SupplierID|Enter Supplier ID|7%>
AND OrderDate >= #<%OrderDate|Order date from|11/1/1994|D%>#
AND Discontinued = <%Discontinued|Is this checked?|true|B%>
```

- Click **OK** to save the data source and return to the report design surface.

The SQL query above causes ActiveReports to display the following dialog to the user. The user can accept these or input other values to select report data.

To understand how this works, see SQL Query Parameters.

Note: Specifying the "param:" prefix for a parameter in the SQL query relates this parameter to the one in the Report Explorer.

For example, "select * from customers where customername = '<%param:Parameter1%>'".

In this case, the parameter with the "param:" prefix in the SQL query is updated with values of the corresponding parameter in the Report Explorer.

SQL Query Parameter Syntax

SQL Query Parameters Syntax

The SQL query above contains three parameters:

- <%SupplierID|Enter Supplier ID|1000%>
- <%OrderDate|Order date|1/1/2001|D%>
- <%Discount|Is this checked?|true|B%>

Each of these parameters follow the syntactical pattern <%**FieldName** | **PromptString** | **DefaultValue** | **Type** | **PromptUser%**>

FieldName

The name of the field you wish to request (e.g. SupplierID or OrderDate). This is the only part of the syntax which is required, so you can use <%**FieldName%**> if you do not wish to use the other values.

Note: Although FieldName is the only required parameter, if you do not specify a DefaultValue for each parameter, the Report Explorer is not populated with bound fields at design time.

PromptString

An optional string value which sets the text that appears in the dialog next to the control (e.g. "Enter Supplier ID: ").

DefaultValue

Sets a default value for the parameter. For example, if you have a date parameter, you can set the `DefaultValue` for the field to the current date so users can just hit ENTER unless they want to generate a report based on a new date. If you do not supply this value, the Report Explorer is not populated with bound fields at design time.

Type

Indicates the type of data requested. The possible values **S** for string, **D** for date, and **B** for Boolean. If you do not provide a value, string is assumed.

A string type provides a textbox for input, a date type provides a calendar drop-down control for input, and a Boolean type provides a check box for input.

PromptUser

Sets whether to prompt the user for a value. The value can be set to **True** for some parameters and **False** for others. If you set the report's `ShowParameterUI` property to **False**, users are not prompted for any parameters, regardless of the `PromptUser` value set for any parameter in the report.



Tip: For Strings, if you specify a default value that is enclosed in apostrophes or quotation marks, ActiveReports sends the same marks to SQL. For Boolean parameters, if you specify true/false for the `DefaultValue` it generates true/false for SQL output. If you specify 0,1, it generates 0 or 1. For date values, enclose the parameter syntax in pound signs, for example, `#<%Date%>#`

Stored Procedures

You can use stored procedures with parameters in ActiveReports. The SQL query has the stored procedure call and placeholders for the parameters: `CustOrderHist <%ID | Enter Customer ID | AFLKI%>`.

ActiveReports replaces the parameter with what the user types into the dialog to create a query like this:

`CustOrderHist 'AFLKI'`.

Run-time Parameters

You can add, edit, and delete parameters at run time. The following code demonstrates how to add a parameter and display its value in an ActiveReports textbox control.

To add parameters at run time in Visual Basic.NET

1. Double-click in the gray area below the report to create an event-handling method for the **ReportStart** event.
2. Add code to the handler to add the parameter at run time.

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Dim myParam1 As New Parameter()
myParam1.Key = "myParam1"
myParam1.Type = Parameter.DataType.String
myParam1.PromptUser = True 'set to False if you do not want input from user
myParam1.Prompt = "Enter last name:"
myParam1.DefaultValue = "This is myParam1 default value"
Me.Parameters.Add(myParam1)
```

3. Back in design view, click in the gray area below the report to select it.
4. Click the events icon in the Properties window to display available events for the report.
5. Double-click `FetchData`. This creates an event-handling method for the report's `FetchData` event.
6. Add code to the handler to pass the parameter at run time.

Visual Basic.NET code. Paste INSIDE the FetchData event.

```
'Set textbox text equal to the value of the parameter
Me.txtParam1.Text = Me.Parameters("myParam1").Value
```

To add parameters at run time in C#

1. Double-click in the gray area below the report to create an event-handling method for the **ReportStart** event.

2. Add code to the handler to add the parameter at run time.

C# code. Paste INSIDE the ReportStart event.

```
Parameter myParam1 = new Parameter();
myParam1.Key = "myParam1";
myParam1.Type = Parameter.DataType.String;
myParam1.PromptUser = true; //set to false if you do not want input from user
myParam1.Prompt = "Enter last name:";
this.Parameters.Add(myParam1);
```

3. Back in design view, click in the gray area below the report to select it.
4. Click the events icon in the Properties window to display available events for the report.
5. Double-click FetchData. This creates an event-handling method for the report's FetchData event.
6. Add code to the handler to pass the parameter at run time.

C# code. Paste INSIDE the FetchData event.

```
//Set textbox text equal to the value of the parameter
this.txtParam1.Text = this.Parameters["myParam1"].Value;
```

Parameters with Subreports

To use parameters with subreports

Add a parameter to the datasource of the subreport as above, either in the sql statement or in code, and ensure that the parameter value exists in the parent report's data. the parameter is passed automatically. The only other consideration is that you need to set the ShowParameterUI property of the subreport to False to prevent the subreport from requesting the parameter value from the user.

You can use parameters with subreports to connect the subreport to the parent report. If you set a parameter for the field that links the parent report to the child subreport, the parent report passes the information to the child through the parameters. Keep the following in mind when working with subreports and parameters:

- Set the subreport's *ShowParametersUI* property to False.
- Set the subreport's SQL query to use the parameter syntax = <%fieldname%>.

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).

Parameters in Charts

To set a parameter in a chart data source




Caution: If you don't set the same ORDER in both SQL queries, that of the report and that of the chart, the chart data is not ordered.

1. With the chart control highlighted, click the **Data Source** verb below the Properties Window to open the Chart DataSource dialog. If you do not see the verb, see the [Access the Chart Wizard and Data Source](#) topic for more information.
2. On the **OLE DB** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis button to browse to the NorthWind database. Click **Open** once you have selected the file.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter a SQL query to select the data that you want.

SQL Query. Paste in the Query field.

```
SELECT * FROM Products WHERE CategoryID = <%CategoryID||1%> ORDER BY ProductName
```


7. Click **OK** to save the data source and return to the report design surface.

 **Note:** To see the chart draw at design time when using parameters, provide a default value. Otherwise, you must run the project and display the report in the viewer in order to see the chart.

Embed Subreports in a Report

To embed a subreport into a parent report, you add two reports to a Visual Studio project, and from the ActiveReports toolbox, drag the SubReport control onto one of the reports. You can then add code to create an instance of the child report, and to display it in the SubReport control.

To add code to create an instance of the child report

 **Warning:** Do not create a new instance of the subreport in the **Format** event. Doing so creates a new subreport each time the section Format code is run, which uses a lot of memory.

To write the code in Visual Basic

1. At the top left of the code view for the parent report, click the drop-down arrow and select **(*rptYourParentReportName* Events)**.
2. 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.
3. Add code to the handler to create a new instance of the child report.

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

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim rpt As rptYourChildReportName
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
rpt = New rptYourChildReportName()
```

To write the code in C#

1. Click in the gray area below the parent report to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportStart**. This creates an event-handling method for the report's ReportStart event.
4. Add code to the handler to create a new instance of the child report.

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

C# code. Paste JUST ABOVE the ReportStart event.

```
private rptYourChildReportName rpt;
```

C# code. Paste INSIDE the ReportStart event.

```
rpt = new rptYourChildReportName();
```

To add code to display the child report in a subreport control on a parent report

1. Double-click in the detail section of the design surface of the parent report to create a detail_Format event.
2. Add code to the handler to display a report in the subreport control.

To write the code in Visual Basic

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

Visual Basic.NET code. Paste INSIDE the Format event.

```
Me.SubReport1.Report = rpt
```

To write the code in C#

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

C# code. Paste INSIDE the Format event.

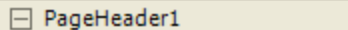
```
this.subReport1.Report = rpt;
```

Pass Parameters to a Subreport

To pass parameters to a subreport, you set the datasources of both reports with the parent report supplying the parameter value for the subreport. There are also some properties that to set on the child report to optimize it to run as a subreport.

To connect the parent report to a data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1




Detail1

2. On the "OLE DB" tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

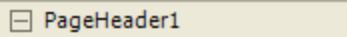
```
SELECT Employees.EmployeeID, Employees.LastName, Employees.FirstName,
Employees.Extension, Customers.CustomerID
FROM Customers, Employees
ORDER BY Employees.EmployeeID, Customers.CustomerID
```

7. Click **OK** to save the data source and return to the report design surface.

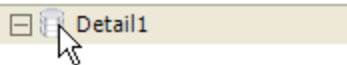
 **Note:** This query joins the Employees table for the parent report to the Customers table for the child report.

To connect the child report to a data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1




Detail1

2. On the "OLE DB" tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

```
SELECT Customers.*, Employees.EmployeeID, Orders.OrderID
FROM Employees INNER JOIN (Customers INNER JOIN Orders ON Customers.CustomerID =
Orders.CustomerID) ON Employees.EmployeeID = Orders.EmployeeID
WHERE CustomerID = '<%CustomerID%>'
```

7. Click **OK** to save the data source and return to the report design surface.

 **Note:** This SQL query uses parameters syntax: '<%CustomerID%>'. For more information on parameters, see the **Parameters** topic.

To set up the child report to be used in the subreport control

1. Click in the gray area below the child report to select the report.
2. In the Properties Window, change the **ShowParametersUI** property to **False**. This prevents the subreport from requesting parameter values from the user.
3. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.

To display the child report in the subreport control on the parent report, see the **Embed Subreports in a Report** topic.

Save and Load Report Files (RDF)

ActiveReports allows reports to be saved into their own standard format called an RDF file (Report Document Format). In this format, the data is static. The saved report displays the data that is retrieved when you run the report. Once a report has been saved to an RDF file, it can be loaded into the viewer control.

To save a report as a static RDF file

1. Double-click the title bar of the Windows Form to create a Form Load event.
2. Add the following code to run and save the report.

To save the report to RDF format in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form load event.

```
Dim rpt As New rptYourReportName()
rpt.Run()
rpt.Document.Save(Application.StartupPath + "\\NewRDF.RDF")
```

To save the report to RDF format in C#**C# code. Paste INSIDE the Form load event.**

```
ActiveReport1 rpt = new ActiveReport1();
rpt.Run();
rpt.Document.Save(Application.StartupPath + "\\NewRDF.RDF");
```

To load a saved RDF file into the ActiveReports viewer

The Windows Form Viewer can display RDF files made with any version of ActiveReports, including COM versions. The FlashViewer viewer type of the WebViewer (Professional Edition) may be able to display RDF files made with previous versions, but this is not guaranteed for every RDF.

1. Double-click the title bar of the Windows Form to create a Form Load event.
2. Add the following code to load the saved report.

To load an RDF file in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Form load event.**

```
Viewer1.Document.Load(Application.StartupPath + "\\NewRDF.RDF")
```

To load an RDF file in C#**C# code. Paste INSIDE the Form load event.**

```
viewer1.Document.Load(Application.StartupPath + "\\NewRDF.RDF");
```

To save or load report files to a memory stream

1. Double-click the title bar of the Windows Form to create an event-handling method for the Form_Load event.
2. Add code to the handler to save the report to a memory stream and load the memory stream into the ActiveReports viewer.

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Form Load event.**

```
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)
```

To write the code in C#**C# code. Paste INSIDE the Form Load event.**

```
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);
```

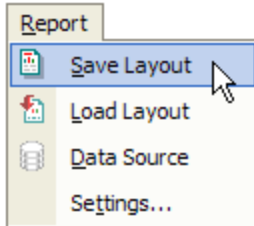
Save and Load Report Layout Files (RPX)

Although ActiveReports writes report layouts in either C# or Visual Basic.NET, you can save the layout of your report as a report XML (RPX) file for portability. If you make changes to the RPX file and load it back into an ActiveReport in Visual Studio, you can see the changes you made reflected in the C# or Visual Basic.NET code in the *YourReportName.Designer.vb* or *YourReportName.Designer.cs* file.

Caution: When you load an RPX layout into a report object, it overwrites everything in the report object. In order to avoid overwriting important layouts, create a new blank ActiveReport and load the RPX file into the new report.

To save a report as an RPX file at design time

1. In the design view of an ActiveReport, from the **Report** menu select **Save Layout**.

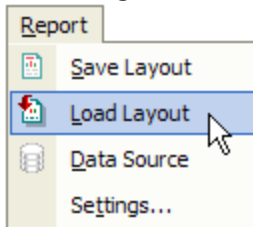


2. Name the file and select the location in which to save it. The file extension is ***.rpx**.
3. Click the **Save** button to save the report layout.

Note: When you save a layout that contains a dataset, ActiveReports saves the data adapter and data connection in the component tray, but not the dataset itself. When the saved layout is loaded into another report, you can regenerate the dataset with the data adapter and data connection.

To load an RPX file at design time


1. In the design view of a new, blank ActiveReport, from the **Report** menu select **Load Layout**.



2. Navigate to the RPX file that you want to load and select it.
3. Click the **Open** button to load the report layout.

To save a report as an RPX file at run time

Save report layouts before they run. If you save a layout after the report runs, you also save any dynamic changes made to properties or sections in the report. To avoid this when you call `SaveLayout` inside the report code, use the `ReportStart` event.

 **Important:** When you save a report layout, ActiveReports only saves the code in the script editor to the file. Any code behind the report in the .cs or .vb file is not saved to the RPX file. For more information, see the **Add Code to Layouts Using Script**

1. Right-click on the Windows Form and select **View Code** from the shortcut menu to see the code view for the Windows form.
2. Add code to the class to save the report.

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET


Visual Basic.NET code. Paste INSIDE the Form class.

```
Private Sub SaveRPX()  
    Dim rpt As New DataDynamics.ActiveReports.ActiveReport()  
    rpt.Run()  
    rpt.SaveLayout("C:\NewRPX.RPX")  
End Sub
```

To write the code in C#

C# code. Paste INSIDE the Form class.

```
private void SaveRPX()  
{  
    DataDynamics.ActiveReports.ActiveReport rpt = new DataDynamics.ActiveReports.ActiveReport();  
    rpt.Run();  
    rpt.SaveLayout(@"C:\NewRPX.RPX");  
}
```

 **Note:** The SaveLayout method uses utf-16 encoding when you save to a stream, and utf-8 encoding when you save to a file.

To load a saved RPX file into the ActiveReports viewer

1. Right-click on the Windows Form to see the code view for the Windows form.
2. Add the following code to the form class to load a report.

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form class.

```
Dim rpt As New DataDynamics.ActiveReports.ActiveReport()  
Private Sub LoadRPX()  
    rpt.LoadLayout("C:\NewRPX.RPX")  
    Viewer1.Document = rpt.Document  
    rpt.Run()  
End Sub
```

To write the code in C#

C# code. Paste INSIDE the Form class.

```
private void LoadRPX()  
{  
    DataDynamics.ActiveReports.ActiveReport rpt = new  
DataDynamics.ActiveReports.ActiveReport();
```



```

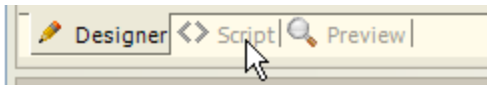
rpt.LoadLayout(@"C:\NewRPX.rpx");
viewer1.Document = rpt.Document;
rpt.Run();
}

```

Add Code to Layouts Using Script

When you save an ActiveReport to Report XML (RPX) format, the RPX file contains the layout, but does not contain any of the Visual Basic or C# code that you may have added to the code behind the report. For this reason, ActiveReports provides a Script tab at the bottom of the report design window.

To access the script editor, click the script tab.




You can use scripting to provide VB.NET or C# functionality to reports without compiling .vb or .cs files. This permits reports saved to report RPX format to serve as stand-alone reports. By including scripting when you save the report as an RPX file, you can later load, run, and display it in the viewer control without the designer. This allows you to update distributed reports without recompiling. You can use Visual Basic or C# script.

ActiveReports loads RPX files, including any scripting, in the InitializeComponent() method.


You can add script to the script editor at design time, or use the rpt.Script property to add it at run time and save it to the RPX file.

Since the RPX file can be read with any text editor, use the AddCode or AddNamedItem method to add secure information, such as a connection string, to a project.

 **Note:** The ActiveReports script editor supports IntelliSense that helps the writing of code by making the access to the language elements fast and easy.

Tips for Using Scripting

- **Make sure the report class is public.** If the report class is not set to public, the script will not be able to recognize the items in your report. The report class is public by default.
- **Make sure the control being referenced in the script has its Modifiers property set to Public.** If the control's Modifiers property is not set to Public, the control cannot be referenced in script and an error will be raised when the report is run. The Modifiers property has a default value of Private, so you must set this property in the designer.
- **Use "this" (as in C# code-behind) or "Me" (as in VB code-behind) to reference the report.** Using "rpt" to reference the report is also possible but it is recommended to use the "this" and "Me" keywords.

 **Note:** The basic approach of using the "this/Me" and "rpt" keywords is as follows - use "this/Me" to access the properties and controls added to the sections of the report, whereas use "rpt" within the instance of the ActiveReports class only to access its public properties, public events and public methods.

- **Use error handling.** When working with scripting, use error handling around the .Run() call. When errors are raised, the returned error should point to the section of script causing the error.
- **Remember to save the layout after you make changes to the report.** It is easy to forget to save the layout after you have made changes to the report in the designer or script editor.

To select the scripting language to use

1. In design view of the report, click in the grey area below the report to select it.
2. In the Properties window, drop down the **ScriptLanguage** property and select **C#** or **VB.NET**.

To use the Script view of the report

When you select the Script tab, there are two drop-down boxes at the top of the tab:


- **Object** Drop down the list and select one of the report sections, or the report itself.
- **Event** If you select a report section as the Object, there are three events: Format, BeforePrint, and AfterPrint. For more information, see the **Section Events** topic. If you select ActiveReport as the Object, there are seven events. For more information, see the **Report Events** topic. Select an event to create an event handling method in the scripting language you chose for the report.

Add script to the events in the same way that you add code to events in the code view of the report.

To access controls in script

To add script to a report to access a textbox named TextBox1 in the detail section and assign the text "Hello" to it:

1. On the script tab of the report, drop down the **Object** list and select **Detail1**. This populates the Event drop-down list with section events.
2. Drop down the **Event** list and select **Format**. This creates script stubs for the event.

 **Note:** Use the examples with the "this" and "Me" keywords, as they are recommended rather than the ones with "rpt".

To access a textbox in the detail section in VB.NET script

Visual Basic.NET script. On the Script tab of the report, paste INSIDE the Detail Format event.

```
Me.TextBox1.Text = "Hello"
```

Visual Basic.NET script. On the Script tab of the report, paste INSIDE the Detail Format event.

```
CType(rpt.Sections("Detail1").Controls("TextBox1"), TextBox).Text = "Hello"
```

To access a textbox in the detail section in C# script

C# script. On the Script tab of the report, paste INSIDE the Detail Format event.

```
this.TextBox1.Text = "Hello";
```

C# script. On the Script tab of the report, paste INSIDE the Detail Format event.

```
((TextBox)rpt.Sections["detail1"].Controls["TextBox1"]).Text = "Hello";
```

To give a script access to functions in a class in your project

Use the **AddNamedItem** method to allow the script to access functions in a class file within your project. This allows you to keep secure information such as a database connection string or a SQL query string in the code instead of in the RPX file.

1. Add a class to your project named **clsMyItem**.
2. Add a public function to your class using code like the following:

To create a public function in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the new class.**

```
Public Function getMyItem() As String
    getMyItem = "Hello"
End Function
```

To create a public function in C#**C# code. Paste INSIDE the new class.**

```
public string getMyItem()
{
    return "Hello";
}
```


3. Open the **ReportStart** event handler (code-behind) and add the following code:
To access the class in VB.NET

VB.NET code. Paste before or in the ReportStart event.


```
Me.AddNamedItem("myItem", new clsMyItem())
```

To access the class in C#**C# code. Paste before or in the ReportStart event.**

```
this.AddNamedItem("myItem", new clsMyItem());
```

 You can also use the AddNamedItem method before the report Run method.

4. On the script tab of the report, drop down the **Object** list and select **Detail1**. This populates the Event drop-down list with section events.
5. Drop down the **Event** list and select **Format**. This creates script stubs for the event.
6. Add script to the event to typecast a control on the report and populate it using the named item.

 **Note:** Use the examples with the "this" and "Me" keywords, as they are recommended rather than the ones with "rpt".

To typecast the control in VB.NET script**VB.NET script. Paste INSIDE the Detail Format event.**

```
Me.textBox1.Text = myItem.getMyItem()
```

VB.NET script. Paste INSIDE the Detail Format event.

```
CType(rpt.Sections("Detail1").Controls("TextBox1"), TextBox).Text =
myItem.getMyItem()
```

To typecast the control in C# script**C# script. Paste INSIDE the Detail Format event.**

```
this.textBox1.Text = myItem.getMyItem();
```

C# script. Paste INSIDE the Detail Format event.

```
((TextBox) rpt.Sections["detail1"].Controls["TextBox1"]).Text = myItem.getMyItem()
```

To access assemblies

Use the **AddScriptReference** method to gain access to .NET or other assemblies. This is only necessary if you need a reference, such as System.Data.dll, that is not initialized in the project before the script runs.

To access an assembly in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form code.

```
Private Sub runReport()  
    Dim rpt as new ActiveReport1()  
    rpt.AddScriptReference("System.Data.dll")  
    rpt.Run()  
    Me.Viewer1.Document = rpt.Document  
End Sub
```

To access an assembly in C#

C# code. Paste INSIDE the Form code.

```
private void runReport()  
{  
    ActiveReport1 rpt = new ActiveReport1();  
    rpt.AddScriptReference("System.Data.dll");  
    rpt.Run();  
    this.viewer1.Document = rpt.Document;  
}
```

To add code to a report's script from a Windows Form

Use the AddCode method to inject code into the script.

The AddCode method allows you to add actual code segments to the script at run time. This is useful for allowing secure information, such as a database connection string or SQL query string, to be used inside the script without saving it into the RPX file.

To add code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the report class.

```
Public Function addThisCode() As String  
    Dim sCode As String = "Public Function ShowACMessage() As String" +  
Environment.NewLine + "ShowACMessage = ""my Added Code"" + Environment.NewLine + "End  
Function"  
    addThisCode = sCode  
End Function
```

VB.NET script. Paste INSIDE the ReportStart event.

```
rpt.AddCode(addThisCode)
```

To add code in C#


C# code. Paste INSIDE the report class.

```
public string addThisCode()  
{
```

```
string sCode = "public string ShowACMessage() {return \"my Added Code\";}";
return sCode;
}
```

C# script. Paste INSIDE the ReportStart event.

```
rpt.AddCode(addThisCode());
```

 **Note:** Use the examples with the "this" and "Me" keywords, as they are recommended rather than the ones with "rpt".

To write the script in Visual Basic.NET

VB.NET script. Paste INSIDE the Detail Format event.

```
Me.TextBox1.Text = ShowACMessage()
```

VB.NET script. Paste INSIDE the Detail Format event.

```
CType(rpt.Sections("Detail1").Controls("TextBox1"), TextBox).Text = ShowACMessage()
```

To write the script in C#

C# script. Paste INSIDE the Detail Format event.

```
this.TextBox1.Text = ShowACMessage();
```

C# script. Paste INSIDE the Detail Format event.

```
((TextBox)rpt.Sections["Detail1"].Controls["TextBox1"]).Text = ShowACMessage();
```

To create classes inside the script to call methods

If the script requires method calls, you can construct a class inside the script.

 **Note:** Use the examples with the "this" and "Me" keywords, as they are recommended rather than the ones with "rpt".

To create a class inside the script in VB.NET script

VB.NET script. Paste INSIDE the report class.

```
Public Class MyFuncs
    Public Sub New()
    End Sub
    Public Function ShowMyString() As String
        Return "This is my string"
    End Function
End Class
```

VB.NET script. Paste INSIDE the Detail Format event.

```
Dim f As MyFuncs = New MyFuncs()
Me.TextBox1.Text = f.ShowMyString
```

VB.NET script. Paste INSIDE the Detail Format event.

```
Dim f As MyFuncs = New MyFuncs()
CType(rpt.Sections("Detail").Controls("TextBox1"), TextBox).Text = f.ShowMyString
```

To create a class inside the script in C#

C# script. Paste INSIDE the report class.

```
public class MyFuncs
{
    public MyFuncs()
    {
    }
    public string ShowMyString()
    {
        return "This is my string";
    }
}
```

C# script. Paste INSIDE the Detail Format event.

```
MyFuncs f = new MyFuncs();
this.TextBox1.Text = f.ShowMyString();
```

C# script. Paste INSIDE the Detail Format event.


```
MyFuncs f = new MyFuncs();
((TextBox)rpt.Sections["Detail1"].Controls["TextBox1"]).Text = f.ShowMyString();
```

Provide One-Touch Printing in the WebViewer (Pro Edition)

Professional Edition licensees can provide one-touch printing using the new FlashViewer ViewerType of the WebViewer control.


To provide one-touch printing

1. From the Visual Studio toolbox, drag the WebViewer control onto your ASPX page. (See **Adding ActiveReports Controls** for help if you have not yet added the control to your toolbox.)
2. Add an ActiveReport to the project.
3. Back on the ASPX page, click to select the WebViewer, and in the Properties window, make the following changes:
 - a. Set the **ReportName** property to the name of your report.
 - b. Set the **ViewerType** property to **FlashViewer**.
 - c. Expand the **FlashViewerOptions** and **PrintOptions** nodes, and set the **StartPrint** property to **True**.
 - d. If you do not want to display the report to the user, set the **Height** and **Width** properties to **0**.
4. Copy the **ActiveRepors.FlashViewer.swf** file into your project folder. This file is located in: **C:\Program Files\GrapeCity\ActiveReports 6\Deployment** (on a **64-bit Windows operating system**, this file is located in **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment**).
5. Run the project to see the **Print** dialog box and send the report to print by clicking **Print** in the dialog box.

 **Note:** The **Print** dialog box will appear at least once because of the Flash printing limitations.

Provide One-Touch Printing in Web Applications (Pro Edition)


With the one-touch PDF printing, you can have the PDF output open in the browser with the **Print** dialog.

 **Note:** To learn how to provide one-touch printing in the PDF output with Standard Edition, see **Custom Web Exporting ('Custom Web Exporting (Std Edition)' in the on-line documentation).**

1. From the Visual Studio **File** menu, select **New Project...**
2. In the New Project dialog that appears, select **ASP.NET Web Application** (If you have the .NET Framework 4, in the **Visual Studio 2010** New Project dialog that appears, select **ASP.NET Empty Web Application**).
3. Rename the project and click the **OK** button.
4. (For Visual Studio 2010 .NET Framework 4 only) From the Visual Studio 2010 **Project** menu, select **Add New Item**.
5. (For Visual Studio 2010 .NET Framework 4 only) Select **Web Form**, rename it to **Default.aspx** and click **Add**.
6. (For Visual Studio 2010 .NET Framework 4 only) In the Solution Explorer, right-click the Web Form that you have added and select **Set As Start Page**.
7. From the **Project** menu, select **Add New Item**.
8. Select **ActiveReports 6 (xml-based) File**, rename to **MyReport.rpx**, and click **Add**.
9. In the Solution Explorer, right-click the **References** folder and select **Add Reference**.
10. In the dialog that appears, go to **.NET**, select **ActiveReports.Document.dll**, **ActiveReports6.dll**, **ActiveReports.PdfExport.dll**, **ActiveReports.HtmlExport.dll** and **ActiveReports.Web.dll**, and click **OK**.
11. In the Solution Explorer, open Web.config and add http handlers as follows.

XML code. Paste in the XML view of the Web.config file inside the system.web section.

```
<httpHandlers>
  <add verb="*" path="*.rdf.RdfItem"
  type="DataDynamics.ActiveReports.Web.Handlers.RdfHandler, ActiveReports.Web,
  Version=6.2.4209.0, Culture=neutral, PublicKeyToken=cc496777c49a3ff"/>
  <add verb="*" path="*.rpx"
  type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
  Version=6.2.4209.0, Culture=neutral, PublicKeyToken=cc496777c49a3ff"/>
  <add verb="*" path="*.ActiveReport"
  type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler,
  ActiveReports.Web, Version=6.2.4209.0, Culture=neutral,
  PublicKeyToken=cc496777c49a3ff"/>
  <add verb="*" path="*.ArCacheItem"
  type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler,
  ActiveReports.Web, Version=6.2.4209.0, Culture=neutral,
  PublicKeyToken=cc496777c49a3ff"/>
</httpHandlers>
```

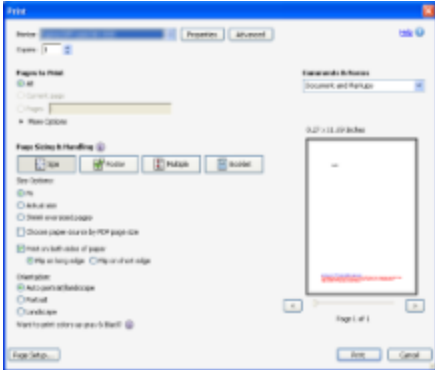
 **Note:** You should update the Version and PublicKeyToken values to reflect the current version of ActiveReports installed on your machine. You can find the Version and PublicKeyToken values in the Global Assembly Cache (GAC), C:\WINDOWS\ASSEMBLY.

12. On the Default.aspx page, add the following code to .aspx source.

Paste into .aspx source to the body section between the div tags

```
<a href="MyReport.rpx?OutputFormat=PDF&Print=true" >Pdf print of rpx file</a>
```

13. Run the project.
14. On the Default page that opens in the browser, click the link **Pdf print of rpx file** to open the PDF file with the **Print** dialog.



Note: For an rdf file, your link may look similar to the following.

```
<a href="PdfPrintTest.rdf.RdfItem?OutputFormat=PDF&Print=true" >Pdf print of rdf file</a>
```

Add Designer ToolStrips (Pro Edition)

The Designer control has built-in methods to add its menus and toolbars to a Visual Studio ToolStripContainer in your end user designer project.

To add Designer ToolStrips

Prior to following these steps, add an ActiveReports Designer control and a Visual Studio ToolStripContainer to your form.



If you need help adding the Designer control to your Visual Studio toolbox, see **Adding ActiveReports Controls**.

1. Double-click the designer control on the form to create the **Designer Load** event.
2. Add code like the following to the event:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Load event.

```
Dim menuStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Menu)
(0)

Dim editStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Edit)
(0)

Dim formatStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Format)
(0)

Dim layoutStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Layout)
(0)

Dim reportStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Report)
(0)

Dim undoStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Undo)
(0)

Dim zoomStrip As ToolStrip =
Me.Designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Zoom)
(0)

Me.ToolStripContainer1.TopToolStripPanel.Join(menuStrip, 0)
Me.ToolStripContainer1.TopToolStripPanel.Join(editStrip, 1)
Me.ToolStripContainer1.TopToolStripPanel.Join(formatStrip, 2)
Me.ToolStripContainer1.TopToolStripPanel.Join(layoutStrip, 3)
Me.ToolStripContainer1.TopToolStripPanel.Join(reportStrip, 4)
```



```
Me.ToolStripContainer1.TopToolStripPanel.Join(undoStrip, 5)
Me.ToolStripContainer1.TopToolStripPanel.Join(zoomStrip, 6)
```

To write the code in C#

C# code. Paste INSIDE the Load event.

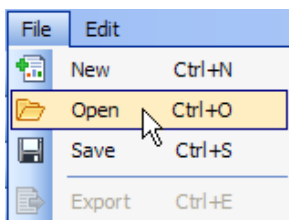
```
ToolStrip menuStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Menu)
[0];
ToolStrip editStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Edit)
[0];
ToolStrip formatStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Format)
[0];
ToolStrip layoutStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Layout)
[0];
ToolStrip reportStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Report)
[0];
ToolStrip undoStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Undo)
[0];
ToolStrip zoomStrip =
this.designer1.CreateToolStrips(DataDynamics.ActiveReports.Design.DesignerToolStrips.Zoom)
[0];
this.toolStripContainer1.TopToolStripPanel.Join(menuStrip, 0);
this.toolStripContainer1.TopToolStripPanel.Join(editStrip, 1);
this.toolStripContainer1.TopToolStripPanel.Join(formatStrip, 2);
this.toolStripContainer1.TopToolStripPanel.Join(layoutStrip, 3);
this.toolStripContainer1.TopToolStripPanel.Join(reportStrip, 4);
this.toolStripContainer1.TopToolStripPanel.Join(undoStrip, 5);
this.toolStripContainer1.TopToolStripPanel.Join(zoomStrip, 6);
```

- See the table below for the run-time results of adding each tool strip.

ToolStrips at Run Time

ToolStrip Run-Time Name

Menu



The File menu includes these commands: New, Open, Save, and Export.

The Edit menu includes these commands: Undo, Redo, Cut, Copy, Paste, Delete, and Select All.

Edit



The Edit tool strip includes these tools: Cut, Copy, Paste, and Delete.

Format



The Format tool strip includes these tools: Font Name, Font Size, Bold, Italic, Underline, Fore Color, Back Color, Align Left, Align Center, Align Right, Align Justify, Indent, and Outdent.

Layout



The Layout tool strip includes these tools: Align to Grid, Align Lefts, Align Centers, Align Rights, Align Tops, Align Middles, Align Bottoms, Bring to Front, and Send to Back.

Report



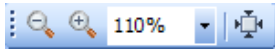
The Report tool strip includes these tools: New, Open, and Save.

Undo



The Undo tool strip includes these tools: Undo and Redo.

Zoom



The Zoom tool strip includes these tools: Zoom Out, Zoom In, Zoom %, and Actual Size.

Add Report Links to Web Forms (Pro Edition)

To add hyperlinks to reports, you must first configure the ActiveReports HttpHandlers and save your report layouts to report XML (RPX) format.

For more information, see **Configure HTTPHandlers (Pro Edition)** and **Save and Load Report Layout Files (RPX)**.

To enable the HTTPHandlers

1. In the Solution Explorer, double-click the **Web.config** file.
2. In the XML view, add the following code to the Web.config file.

HttpHandler Configuration XML

XML code. Paste in the XML view of the Web.config file inside the system.web section.

```
<httpHandlers>
  <!-- ***** ActiveReports HttpHandler Configuration ***** -->
  <add verb="" path="*.rpx"
type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
Version=6.0.865.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
  <add verb="" path="*.ActiveReport"
type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler,
ActiveReports.Web, Version=6.0.865.0, Culture=neutral,
PublicKeyToken=cc4967777c49a3ff" />
  <add verb="" path="*.ArCacheItem"
type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler,
ActiveReports.Web, Version=6.0.865.0, Culture=neutral,
PublicKeyToken=cc4967777c49a3ff" />
</httpHandlers>
```

3. Update the **Version** and **PublicKeyToken** values to reflect the current version of ActiveReports installed on your machine.



Tip: You can find the Version and PublicKeyToken values in the Global Assembly Cache (GAC), C:\WINDOWS\ASSEMBLY.

To add a link to a Web Form that opens a report in PDF format

1. In the HTML view of the Web Form (*.aspx), add a hyperlink like the following.

Hyperlink HTML

HTML code. Paste in the HTML view of the Web Form.

```
<a href="rptYourReportName.rpx?OutputFormat=pdf">Open the report in PDF format.</a>
```

2. Press **F5** to run the program.
3. Click the link on the web form to view the report PDF.

To add a link that passes a parameter to a report and opens it in HTML format

In order to pass a parameter to a report, you must first **Add Parameters** to the report. In this case, we are using a report with a *Country* parameter.



Caution: Set your report's **ShowParameterUI** property to **False** to prevent the server from hanging while it tries to show the parameter dialog box.

Tip: Remember to save your report layout to RPX format again after you make any changes.

1. In the HTML view of the Web Form, add a hyperlink like the following.

Hyperlink HTML

HTML code. Paste in the HTML view of the Web Form.

```
<a href="rptYourReportName.rpx?Country=USA">Customer Phone List for USA</a>
```

2. Press **F5** to run the program.
3. Click the link on the web form to show the report in its default HTML format.

Provide PDF Printing in the Silverlight Viewer (Pro Edition)

You can provide PDF printing in your Silverlight project, which allows to print a document from Silverlight to the PDF format directly. This is a good alternative to the default Silverlight printing with its large print spool size issue.



Note: The PDF printing only works for reports that are loaded from a static RDF file (except RDF files saved on the client side) and an RPX handler. As for reports loaded from a document stream (aspx page) and WCF service, the PDFPrint property is ignored.

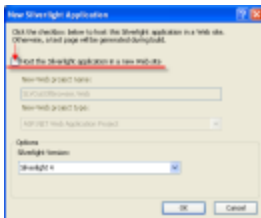
If PDF printing was set up in the Silverlight project, you will see this Print dialog by clicking **Print** in the Silverlight Viewer toolbar:

Use the Silverlight Viewer in Silverlight Out-of-Browser Applications (Pro Edition)

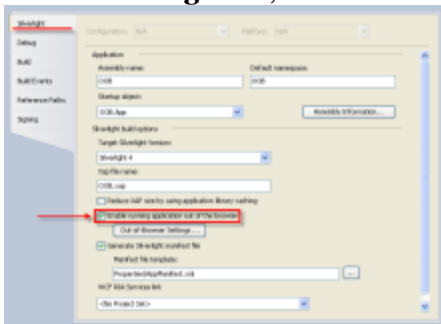
With the Silverlight out-of-browser support, you can configure your Silverlight-based application to run outside a browser.



1. On the Visual Studio **File** menu, click **New Project**.
2. In the **New Project** dialog that appears, select **Silverlight Application** in the list of templates.
3. Specify the name and location of the project and then click the **OK** button.
4. In the **New Silverlight Application** dialog, uncheck **Host the Silverlight application in a new Web site** and then click the **OK** button.



5. From the **Project** menu, select *YourSilverlightProject* Properties.
6. On the **Silverlight** tab, select **Enable Running Application Out Of The Browser**.



7. Add the Silverlight Viewer control as described in the section **To add the Silverlight Viewer control of Silverlight Viewer (Pro Edition)**.
8. Bind a report by using OpenFileDialog as described in the section **To bind a report to the ActiveReports Silverlight Viewer of Silverlight Viewer (Pro Edition)**.
9. Press F5 to run the project.

Make Changes to a Report Undoable in the End User Designer (Pro Edition)

In ActiveReports Professional Edition, you can set up a **custom end-user report designer** on a Windows form. When designing a report in the end-user report designer, you can add a section or an ActiveReports control to a report, remove it from a report and modify the

properties of the ActiveReports **report** ('ActiveReport Class' in the on-line documentation), **section** ('Section Class' in the on-line documentation) and **control** ('ARControl Class' in the on-line documentation) classes in code-behind, thus making the changes undoable. This means that you can undo at run time any modification you have made to an ActiveReports report, section or control property in code-behind.

To add a section

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to add a section (for example, the **ReportHeader** and **ReportFooter** section).

To write the code in Visual Basic

Paste INSIDE the button_Click event

```
Me.arDesigner.ExecuteAction(DesignerAction.InsertReportHF)
```

To write the code in C#

Paste INSIDE the button_Click event

```
this.arDesigner.ExecuteAction(DesignerAction.InsertReportHF);
```

To add an ActiveReports control to the section

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to add an ActiveReports control (for example, the **TextBox** control).

To write the code in Visual Basic

Paste INSIDE the button_Click event

```
Dim control As New TextBox
control.Name = "TextBox1"
control.Size = New SizeF(1.0F, 1.0F)
Me.arDesigner.AddComponent(Me.arDesigner.Report.Sections("Detail1"), control)
```

To write the code in C#

Paste INSIDE the button_Click event

```
TextBox control = new TextBox();
control.Name="TextBox1";
control.Size = new SizeF(1f,1f);
this.arDesigner.AddComponent(this.arDesigner.Report.Sections["Detail1"], control);
```

To add an ActiveReports control to the section at the specified index

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to add an ActiveReports control (for example, the **TextBox** control).

To write the code in Visual Basic

Paste INSIDE the button_Click event

```
Dim control As New TextBox
control.Name = "TextBox1"
control.Size = New SizeF(1.0F, 1.0F)
Me.arDesigner.AddComponent(Me.arDesigner.Report.Sections("Detail1"), control, 0)
```

To write the code in C#

Paste INSIDE the button_Click event

```
TextBox control = new TextBox();
control.Name="TextBox1";
control.Size = new SizeF(1f,1f);
this.arDesigner.AddComponent(this.arDesigner.Report.Sections["Detail1"], control, 0);
```

To remove a section

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to remove a section (for example, the **PageHeader** section).

To write the code in Visual Basic

Paste INSIDE the button_Click event

```
Me.arDesigner.RemoveComponent(Me.arDesigner.Report.Sections("PageHeader1"))
```

To write the code in C#**Paste INSIDE the button_Click event**

```
this.arDesigner.RemoveComponent(this.arDesigner.Report.Sections["PageHeader1"]);
```

To remove an ActiveReports control

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to remove an ActiveReports control (for example, the **TextBox** control).

To write the code in Visual Basic**Paste INSIDE the button_Click event**

```
Me.arDesigner.RemoveComponent(Me.arDesigner.Report.Sections("Detail1").Controls("TextBox1"))
```

To write the code in C#**Paste INSIDE the button_Click event**

```
this.arDesigner.RemoveComponent(this.arDesigner.Report.Sections["Detail1"].Controls["TextBox1"]);
```

To change a report property

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event of a report to change a report property (for example, the **PrintWidth** property).

To write the code in Visual Basic**Paste INSIDE the button_Click event**

```
Me.arDesigner.UpdateComponent(Me.arDesigner.Report, "PrintWidth", 2.0F)
```

To write the code in C#**Paste INSIDE the button_Click event**

```
this.arDesigner.UpdateComponent(this.arDesigner.Report, "PrintWidth", 2.0f);
```

To change a section property

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event of a report to change a section property (for example, the **Height** property).

To write the code in Visual Basic**Paste INSIDE the button_Click event**

```
Me.arDesigner.UpdateComponent(Me.arDesigner.Report.Sections("Detail1"), "Height", 10.0F)
```

To write the code in C#**Paste INSIDE the button_Click event**

```
this.arDesigner.UpdateComponent(this.arDesigner.Report.Sections["Detail1"], "Height", 10.0f);
```

To change an ActiveReports control property

1. Add the **Button** control to the Design view of EndUserDesignerMainForm.
2. Add code to the **button_Click** event to change a control property (for example, the **Text** property).

To write the code in Visual Basic**Paste INSIDE the button_Click event**

```
Me.arDesigner.UpdateComponent(Me.arDesigner.Report.Sections("Detail1").Controls("TextBox1"), "Text", "Test")
```

To write the code in C#**Paste INSIDE the button_Click event**

```
this.arDesigner.UpdateComponent(this.arDesigner.Report.Sections["Detail1"].Controls["TextBox1"], "Text", "Test");
```

Customize, Localize and Deploy

ActiveReports uses an English locale by default, and includes localization resources for Japanese and Russian locales. You can also localize all of the components into any language you need. GrapeCity may, from time to time and on the agreement of users who localize components, include additional locales with future hot fixes and service packs. If you are willing to share your localized resources with other users, please inform technical support staff so that they can pass on your resource files to development.

There are several ways to deploy your ActiveReports applications. See the topics listed below for more information on localizing and deploying your applications.

This section contains information about how to:

Localize ActiveReport Classes, TextBoxes, and Chart Controls

Learn how to localize individual textboxes, chart controls, and entire reports.

Customize the Viewer Control

Learn how to customize the viewer control in a report.

Localize the Viewer Control

Learn how to localize settings for the ActiveReports Viewer control.

Localize Active Reports Resources

Learn to localize ActiveReports dialogs.

Deploy Windows Applications

Learn to deploy ActiveReports Windows applications.

Deploy Web Applications (Std Edition)

Learn to deploy Web applications with the Standard Edition of ActiveReports.

Localize the End User Report Designer

Learn to localize all of the strings in the End User Report Designer.

Deploy the End User Report Designer (Pro Edition)

Learn how to properly deploy the End User Report Designer (Professional Edition).

Localize the Flash Viewer

Learn to localize the FlashViewer in the WebViewer (Professional Edition).

Customize the FlashViewer Toolbar (Pro Edition)

Learn to add, remove, and rearrange buttons in the FlashViewer toolbar.

Deploy Web Applications (Pro Edition)

Learn to deploy Web applications with the WebViewer.

Customize End User Designer Help (Pro Edition)

Learn what third-party tools you need to customize the help project for your designer application.

Deploy End User Designer Help (Pro Edition)

Learn to deploy the help file for your designer application.

Configure HTTPHandlers (Pro Edition)

Learn to configure IIS to use Professional Edition HTTP handlers.

Configure Handler Mappings in IIS 7.x

Learn to configure IIS 7.x to use Professional Edition HTTP handlers, or to deploy without configuring IIS (v7 only).

Customize the Silverlight Viewer (Pro Edition)


Learn how to change the look of the Silverlight Viewer and its elements.

Localize the Silverlight Viewer (Pro Edition)

Learn how to localize the Silverlight Viewer with included resources or create custom localizations.

Localize ActiveReport Classes, TextBoxes, and Chart Controls

Each of the following objects has a public Culture property that allows you to define how ActiveReports formats data when the OutputFormat is set to D (date), C (currency), or other .NET format specifiers.

 **Note:** The default value for the Culture property is **(default, inherit)**.
 For the ActiveReport object, this is the culture of the current thread.
 For the TextBox and ChartControl, this is the culture of the ActiveReport object.

Design Time

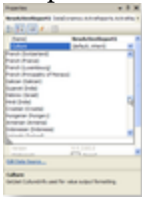
At design time, you can set the globalization culture in the Visual Studio Properties grid.

To localize an ActiveReport at design time

1. In the Design view of the report, select the ActiveReport object in the Properties grid.

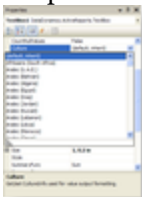


2. Drop down the Culture property list, and select the culture that you want to apply to the report.



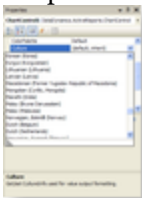
To localize a TextBox at design time

1. In the Design view of the report, select the TextBox control that you want to localize in the Properties grid.
2. Drop down the Culture property list, and select the culture that you want to apply to the textbox.



To localize a Chart at design time

1. In the Design view of the report, select the ChartControl in the Properties grid.
2. Drop down the Culture property list, and select the culture that you want to apply to the chart.



Run Time

You can also specify a culture in code. For a list of System.Globalization culture codes, see **Cultures**.

To localize an ActiveReport at run time

1. In the Design view of the report, double-click in the grey area below the report design surface to create an event handling method for the ReportStart event.
2. In the code view of the report that appears, paste code like the following.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
MyReport.Culture = System.Globalization.CultureInfo.CreateSpecificCulture("en-US")
```

To write the code in C#

C# code. Paste INSIDE the ReportStart event.

```
MyReport.Culture = System.Globalization.CultureInfo.CreateSpecificCulture("en-US");
```

To localize a TextBox at run time

1. In the Design view of the report, double-click the section containing the TextBox control that you want to localize to create an event handling method for the section Format event.
2. In the code view of the report that appears, paste code like the following in the Format event.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
TextBox1.Culture = System.Globalization.CultureInfo.CreateSpecificCulture("en-US")
```

To write the code in C#

C# code. Paste INSIDE the Format event.

```
textBox1.Culture = System.Globalization.CultureInfo.CreateSpecificCulture("en-US");
```

To localize a Chart at run time

1. In the Design view of the report, double-click the section containing the ChartControl that you want to localize to create an event handling method for the section Format event.
2. In the code view of the report that appears, paste code like the following in the Format event.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
ChartControll1.Culture =  
System.Globalization.CultureInfo.CreateSpecificCulture("en-US")
```

To write the code in C#

C# code. Paste INSIDE the Format event.

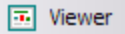
```
chartControll1.Culture =  
System.Globalization.CultureInfo.CreateSpecificCulture("en-US");
```

Customize the Viewer Control

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

To create a basic preview form

1. Open a new Windows Forms project in Visual Studio and size the form according to your needs.
2. From the Visual Studio toolbox, drag the Viewer control onto the form. If you have not added the viewer to the toolbox, see **Adding ActiveReports Controls** for more information.



3. In the Properties window, set the **Dock** property to **Fill**.
4. From the **Project** menu, select **Add New Item**.
5. Select **ActiveReports 6 File** and click the **Add** button.
6. Double-click in the title bar of the form to create a Form Load event.
7. Add the following code to run the report and display the resulting document in the viewer.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt as new NewActiveReport1
rpt.Run()
Viewer1.Document = rpt.Document
```

To write the code in C#

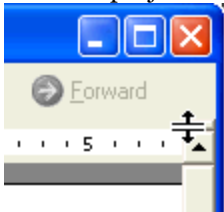
C# code. Paste INSIDE the Form Load event.

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

8. Press **F5** to run the project.

To use split windows on the viewer control

1. With the project running, click the splitter control and drag downward.



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

To add a custom print button to the viewer control

1. Add a second Windows Form to the project created above and name it **frmPrintDlg**.
2. Add a label to **frmPrintDlg** and change the **Text** property to **This is the custom print dialog**.
3. Add a button to **frmPrintDlg** and change the **Text** property to **OK**.
4. Back on the viewer form, double-click the title bar of the form to create a Form Load event.
5. Add the following code to the Form Load event to remove the default print button and add your own.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
' 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)
```

To write the code in C#**C# code. Paste INSIDE the Form Load event.**

```
// 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);
```

6. Add the following code to the Viewer ToolClick event to display **frmPrintDlg** when the custom print button is clicked.

To write the code in Visual Basic.NET

- a. At the top left of the code view of the viewer form, click the drop-down arrow and select **(YourReportName Events)**.
- b. At the top right of the code window, click the drop-down arrow and select ToolClick. This creates an event-handling method for the viewer's ToolClick event.
- c. Add code to the handler to display frmPrintDlg when the custom print button is clicked.

Visual Basic.NET code. Paste INSIDE the Viewer ToolClick event.

```
' 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
```

To write the code in C#

- a. Click the viewer on the form to select it.
- b. Click the events icon in the Properties Window to display available events for the viewer.
- c. Double-click **ToolClick**. This creates an event-handling method for the viewer's ToolClick event.
- d. Add code to the handler to display frmPrintDlg when the custom print button is clicked.

C# code. Paste INSIDE the Viewer ToolClick event.

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

```

        dlg.ShowDialog(this);
    }


```

7. Press **F5** to run the project and see the custom **MyPrint** button on the viewer.


Localize the Viewer Control

To localize the viewer control

1. In Windows Explorer, navigate to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, navigate to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**).
2. Edit the Viewer.bat file:
 - a. Right-click the **Viewer.bat** file and select **Edit**.
 - b. Change the culture in the line **set Culture="en-Us"** to the culture you want to use. For your convenience, here is a list of predefined .NET **Cultures**.
 - c. Ensure that the **ProgramFilesDDPath** is correct.

 **Caution:** Do not change the **ProjectName**, **dllName**, **msDir**, or **BaseNamespace**.

- d. Save and close the Viewer.bat file.
3. Change strings in the resource files:
 - a. Double-click the **Viewer.zip** file to open it.
 - b. Extract all of the files to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, extract the files to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**). A **Viewer** subfolder is created.
 - c. In the new Viewer folder's **Res** subfolder, open each of the four *.resx files and change the strings as needed.
 - d. If you want to change any of the images, rename your localized images to the names of the ones in the **Res\Resources** subfolder and replace them with your localized images.
 4. Back in the main Localization folder, double-click the **Viewer.bat** file to run it. The NameCompleter.exe application runs, and creates:
 - A SatelliteAssembly folder inside the Viewer folder.
 - A language subfolder with the same name as the culture you set in the Viewer.bat file inside the SatelliteAssembly folder.
 - A localized ActiveReports.Viewer6.resources.dll file inside the language subfolder.

 **Note:** The Viewer.bat file copies the language subfolder with the ActiveReports.Viewer6.resource.dll to the **C:\Program Files\Common Files\GrapeCity\ActiveReports 6** folder (on a **64-bit Windows operating system**, the **C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6** folder). If you get a warning message when running the **Viewer.bat** file, then you should use the administrator permissions for the Viewer.bat file.

Run the Viewer.bat file as Administrator

- a. From the **Start** menu, click **Run**, type **cmd** in the text box, and then click **OK**.
- b. Press **CTRL + SHIFT + ENTER** to open a command prompt in the Administrator mode. Alternatively, you can right-click **cmd.exe** and select **Run as Administrator**.
- c. Type **cd C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, type **cd C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**) to change the current directory.
- d. Type **Viewer.bat** to run it.

Log in as an Administrator

The Administrator account is disabled by default so you should activate it first.

- a. From the **Start** menu, click **Run**, type **cmd** in the text box, and then click **OK**.
- b. Press CTRL + SHIFT + ENTER to open a command prompt in the Administrator mode.
- c. Type **net user Administrator /active:yes** to activate the Administrator account.
- d. Log off and log in with the Administrator account.

5. Copy the language subfolder and paste it into the Debug folder of your application.



Note: If you want to put your localization in the Global Assembly Cache (GAC), you must first send the localized ActiveReports.Viewer6.resources.dll file to [GrapeCity](#) and get it signed. Then you can drag the language subfolder with the signed dll file into C:\WINDOWS\ASSEMBLY.

Since the UAC in Vista and Windows7 prevents drag and drop registry to the GAC, we recommend that users of these operating systems run the [Gacutil.exe](#) as described on MSDN.

To test your localized application on a machine that does not share the culture of the localized dll

1. Add the following code in the form's constructor just before the InitializeComponent method is called.
2. Replace the "ja" in the example code with the culture specified in the Viewer.bat file.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = New
System.Globalization.CultureInfo("ja")
```

To write the code in C#

C# code. Paste **INSIDE** the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("ja");
```

Cultures

For your convenience, here is a list of predefined System.Globalization cultures. (Source: [MSDN](#).) For ActiveReports localization purposes, use the Culture/Language Name value in the first column.

| Culture/Language Name | Culture Identifier | Culture |
|-----------------------|--------------------|--------------------------|
| "" (empty string) | 0x007F | Invariant culture |
| af | 0x0036 | Afrikaans |
| af-ZA | 0x0436 | Afrikaans (South Africa) |
| sq | 0x001C | Albanian |
| sq-AL | 0x041C | Albanian (Albania) |
| ar | 0x0001 | Arabic |
| ar-DZ | 0x1401 | Arabic (Algeria) |

| | | |
|------------|--------|------------------------------|
| ar-BH | 0x3C01 | Arabic (Bahrain) |
| ar-EG | 0x0C01 | Arabic (Egypt) |
| ar-IQ | 0x0801 | Arabic (Iraq) |
| ar-JO | 0x2C01 | Arabic (Jordan) |
| ar-KW | 0x3401 | Arabic (Kuwait) |
| ar-LB | 0x3001 | Arabic (Lebanon) |
| ar-LY | 0x1001 | Arabic (Libya) |
| ar-MA | 0x1801 | Arabic (Morocco) |
| ar-OM | 0x2001 | Arabic (Oman) |
| ar-QA | 0x4001 | Arabic (Qatar) |
| ar-SA | 0x0401 | Arabic (Saudi Arabia) |
| ar-SY | 0x2801 | Arabic (Syria) |
| ar-TN | 0x1C01 | Arabic (Tunisia) |
| ar-AE | 0x3801 | Arabic (U.A.E.) |
| ar-YE | 0x2401 | Arabic (Yemen) |
| hy | 0x002B | Armenian |
| hy-AM | 0x042B | Armenian (Armenia) |
| az | 0x002C | Azeri |
| az-Cyrl-AZ | 0x082C | Azeri (Azerbaijan, Cyrillic) |
| az-Latn-AZ | 0x042C | Azeri (Azerbaijan, Latin) |
| eu | 0x002D | Basque |
| eu-ES | 0x042D | Basque (Basque) |
| be | 0x0023 | Belarusian |
| be-BY | 0x0423 | Belarusian (Belarus) |
| bg | 0x0002 | Bulgarian |
| bg-BG | 0x0402 | Bulgarian (Bulgaria) |
| ca | 0x0003 | Catalan |
| ca-ES | 0x0403 | Catalan (Catalan) |
| zh-HK | 0x0C04 | Chinese (Hong Kong SAR, PRC) |
| zh-MO | 0x1404 | Chinese (Macao SAR) |
| zh-CN | 0x0804 | Chinese (PRC) |
| zh-Hans | 0x0004 | Chinese (Simplified) |
| zh-SG | 0x1004 | Chinese (Singapore) |
| zh-TW | 0x0404 | Chinese (Taiwan) |
| zh-Hant | 0x7C04 | Chinese (Traditional) |
| hr | 0x001A | Croatian |
| hr-HR | 0x041A | Croatian (Croatia) |

| | | |
|--------|--------|-------------------------------|
| cs | 0x0005 | Czech |
| cs-CZ | 0x0405 | Czech (Czech Republic) |
| da | 0x0006 | Danish |
| da-DK | 0x0406 | Danish (Denmark) |
| dv | 0x0065 | Divehi |
| dv-MV | 0x0465 | Divehi (Maldives) |
| nl | 0x0013 | Dutch |
| nl-BE | 0x0813 | Dutch (Belgium) |
| nl-NL | 0x0413 | Dutch (Netherlands) |
| en | 0x0009 | English |
| en-AU | 0x0C09 | English (Australia) |
| en-BZ | 0x2809 | English (Belize) |
| en-CA | 0x1009 | English (Canada) |
| en-029 | 0x2409 | English (Caribbean) |
| en-IE | 0x1809 | English (Ireland) |
| en-JM | 0x2009 | English (Jamaica) |
| en-NZ | 0x1409 | English (New Zealand) |
| en-PH | 0x3409 | English (Philippines) |
| en-ZA | 0x1C09 | English (South Africa) |
| en-TT | 0x2C09 | English (Trinidad and Tobago) |
| en-GB | 0x0809 | English (United Kingdom) |
| en-US | 0x0409 | English (United States) |
| en-ZW | 0x3009 | English (Zimbabwe) |
| et | 0x0025 | Estonian |
| et-EE | 0x0425 | Estonian (Estonia) |
| fo | 0x0038 | Faroese |
| fo-FO | 0x0438 | Faroese (Faroe Islands) |
| fa | 0x0029 | Farsi |
| fa-IR | 0x0429 | Farsi (Iran) |
| fi | 0x000B | Finnish |
| fi-FI | 0x040B | Finnish (Finland) |
| fr | 0x000C | French |
| fr-BE | 0x080C | French (Belgium) |
| fr-CA | 0x0C0C | French (Canada) |
| fr-FR | 0x040C | French (France) |
| fr-LU | 0x140C | French (Luxembourg) |
| fr-MC | 0x180C | French (Monaco) |

| | | |
|--------|--------|------------------------|
| fr-CH | 0x100C | French (Switzerland) |
| gl | 0x0056 | Galician |
| gl-ES | 0x0456 | Galician (Spain) |
| ka | 0x0037 | Georgian |
| ka-GE | 0x0437 | Georgian (Georgia) |
| de | 0x0007 | German |
| de-AT | 0x0C07 | German (Austria) |
| de-DE | 0x0407 | German (Germany) |
| de-LI | 0x1407 | German (Liechtenstein) |
| de-LU | 0x1007 | German (Luxembourg) |
| de-CH | 0x0807 | German (Switzerland) |
| el | 0x0008 | Greek |
| el-GR | 0x0408 | Greek (Greece) |
| gu | 0x0047 | Gujarati |
| gu-IN | 0x0447 | Gujarati (India) |
| he | 0x000D | Hebrew |
| he-IL | 0x040D | Hebrew (Israel) |
| hi | 0x0039 | Hindi |
| hi-IN | 0x0439 | Hindi (India) |
| hu | 0x000E | Hungarian |
| hu-HU | 0x040E | Hungarian (Hungary) |
| is | 0x000F | Icelandic |
| is-IS | 0x040F | Icelandic (Iceland) |
| id | 0x0021 | Indonesian |
| id-ID | 0x0421 | Indonesian (Indonesia) |
| it | 0x0010 | Italian |
| it-IT | 0x0410 | Italian (Italy) |
| it-CH | 0x0810 | Italian (Switzerland) |
| ja | 0x0011 | Japanese |
| ja-JP | 0x0411 | Japanese (Japan) |
| kn | 0x004B | Kannada |
| kn-IN | 0x044B | Kannada (India) |
| kk | 0x003F | Kazakh |
| kk-KZ | 0x043F | Kazakh (Kazakhstan) |
| kok | 0x0057 | Konkani |
| kok-IN | 0x0457 | Konkani (India) |
| ko | 0x0012 | Korean |

| | | |
|------------|--------|-------------------------------|
| ko-KR | 0x0412 | Korean (Korea) |
| ky | 0x0040 | Kyrgyz |
| ky-KG | 0x0440 | Kyrgyz (Kyrgyzstan) |
| lv | 0x0026 | Latvian |
| lv-LV | 0x0426 | Latvian (Latvia) |
| lt | 0x0027 | Lithuanian |
| lt-LT | 0x0427 | Lithuanian (Lithuania) |
| mk | 0x002F | Macedonian |
| mk-MK | 0x042F | Macedonian (Macedonia, FYROM) |
| ms | 0x003E | Malay |
| ms-BN | 0x083E | Malay (Brunei Darussalam) |
| ms-MY | 0x043E | Malay (Malaysia) |
| mr | 0x004E | Marathi |
| mr-IN | 0x044E | Marathi (India) |
| mn | 0x0050 | Mongolian |
| mn-MN | 0x0450 | Mongolian (Mongolia) |
| no | 0x0014 | Norwegian |
| nb-NO | 0x0414 | Norwegian (Bokmål, Norway) |
| nn-NO | 0x0814 | Norwegian (Nynorsk, Norway) |
| pl | 0x0015 | Polish |
| pl-PL | 0x0415 | Polish (Poland) |
| pt | 0x0016 | Portuguese |
| pt-BR | 0x0416 | Portuguese (Brazil) |
| pt-PT | 0x0816 | Portuguese (Portugal) |
| pa | 0x0046 | Punjabi |
| pa-IN | 0x0446 | Punjabi (India) |
| ro | 0x0018 | Romanian |
| ro-RO | 0x0418 | Romanian (Romania) |
| ru | 0x0019 | Russian |
| ru-RU | 0x0419 | Russian (Russia) |
| sa | 0x004F | Sanskrit |
| sa-IN | 0x044F | Sanskrit (India) |
| sr-Cyrl-CS | 0x0C1A | Serbian (Serbia, Cyrillic) |
| sr-Latn-CS | 0x081A | Serbian (Serbia, Latin) |
| sk | 0x001B | Slovak |
| sk-SK | 0x041B | Slovak (Slovakia) |
| sl | 0x0024 | Slovenian |


| | | |
|--------------|--------|-----------------------------------|
| sl-SI | 0x0424 | Slovenian (Slovenia) |
| es | 0x000A | Spanish |
| es-AR | 0x2CoA | Spanish (Argentina) |
| es-BO | 0x400A | Spanish (Bolivia) |
| es-CL | 0x340A | Spanish (Chile) |
| es-CO | 0x240A | Spanish (Colombia) |
| es-CR | 0x140A | Spanish (Costa Rica) |
| es-DO | 0x1CoA | Spanish (Dominican Republic) |
| es-EC | 0x300A | Spanish (Ecuador) |
| es-SV | 0x440A | Spanish (El Salvador) |
| es-GT | 0x100A | Spanish (Guatemala) |
| es-HN | 0x480A | Spanish (Honduras) |
| es-MX | 0x080A | Spanish (Mexico) |
| es-NI | 0x4CoA | Spanish (Nicaragua) |
| es-PA | 0x180A | Spanish (Panama) |
| es-PY | 0x3CoA | Spanish (Paraguay) |
| es-PE | 0x280A | Spanish (Peru) |
| es-PR | 0x500A | Spanish (Puerto Rico) |
| es-ES | 0x0CoA | Spanish (Spain) |
| es-ES_tradnl | 0x040A | Spanish (Spain, Traditional Sort) |
| es-UY | 0x380A | Spanish (Uruguay) |
| es-VE | 0x200A | Spanish (Venezuela) |
| sw | 0x0041 | Swahili |
| sw-KE | 0x0441 | Swahili (Kenya) |
| sv | 0x001D | Swedish |
| sv-FI | 0x081D | Swedish (Finland) |
| sv-SE | 0x041D | Swedish (Sweden) |
| syr | 0x005A | Syriac |
| syr-SY | 0x045A | Syriac (Syria) |
| ta | 0x0049 | Tamil |
| ta-IN | 0x0449 | Tamil (India) |
| tt | 0x0044 | Tatar |
| tt-RU | 0x0444 | Tatar (Russia) |
| te | 0x004A | Telugu |
| te-IN | 0x044A | Telugu (India) |
| th | 0x001E | Thai |
| th-TH | 0x041E | Thai (Thailand) |

| | | |
|------------|--------|------------------------------|
| tr | 0x001F | Turkish |
| tr-TR | 0x041F | Turkish (Turkey) |
| uk | 0x0022 | Ukrainian |
| uk-UA | 0x0422 | Ukrainian (Ukraine) |
| ur | 0x0020 | Urdu |
| ur-PK | 0x0420 | Urdu (Pakistan) |
| uz | 0x0043 | Uzbek |
| uz-Cyrl-UZ | 0x0843 | Uzbek (Uzbekistan, Cyrillic) |
| uz-Latn-UZ | 0x0443 | Uzbek (Uzbekistan, Latin) |
| vi | 0x002A | Vietnamese |
| vi-VN | 0x042A | Vietnamese (Vietnam) |


Localize Active Reports Resources

To localize ActiveReports Resources

1. In Windows Explorer, navigate to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, navigate to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**).
2. Edit the *.bat files of any assemblies that you want to localize:
 - a. Right-click the *.bat file and select **Edit**.
 - b. Change the culture in the line **set Culture="en-US"** to the culture that you want to use. For your convenience, here is a list of predefined .NET Cultures.
 - c. Ensure that the **ProgramFilesDDPath** is correct.

 **Caution:** Do not change the **ProjectName**, **dllName**, **msDir**, or **BaseNamespace**.

- d. Save and close the *.bat file.
3. Change strings in the resource files:
 - a. Double-click the *.zip file of the assembly that you want to localize to open it.
 - b. Extract all of the files to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, extract the files to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**). A subfolder with the same name as the zip file is created.
 - c. In the new folder's **Res** subfolder, open each of the *.resx files and change the strings as needed.
 - d. If you want to change any of the images, rename your localized images to the names of the ones in the **Res\Resources** subfolder and replace them with your localized images.
 4. Back in the main Localization folder, double-click the *.bat file to run it. The NameCompleter.exe application runs, and creates:
 - A SatelliteAssembly folder inside the new folder.
 - A language subfolder with the same name as the culture you set in the *.bat file inside the SatelliteAssembly folder.
 - A localized ActiveReports.AssemblyName.resources.dll file inside the language subfolder.
 5. Copy the language subfolder and paste it into the Debug folder of your application.

 **Note:** If you want to put your localization in the Global Assembly Cache (GAC), you must first send the localized ActiveReports.AssemblyName.resources.dll file to [GrapeCity](#) and get it signed. Then you can drag the language subfolder with the signed dll file into C:\WINDOWS\ASSEMBLY.

To test your localized application on a machine that does not share the culture of the localized dll

1. Add the following code in the form's constructor just before the InitializeComponent method is called.
2. Replace the "ja" in the example code with the culture specified in the *.bat file.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = New
System.Globalization.CultureInfo("ja")
```

To write the code in C#

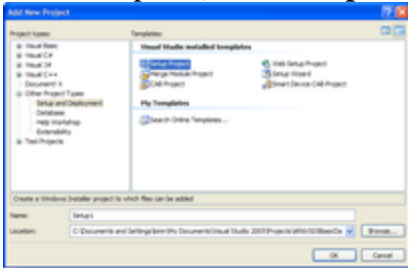
C# code. Paste INSIDE the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("ja");
```

Deploy Windows Applications

To create an installer project

1. Open an existing ActiveReports project or create a new one.
2. From the Visual Studio **Build** menu, select **Build YourActiveReportsProjectName** to build your report project.
3. From the **File** menu, select **Add**, then **New Project** to open the **Add New Project** dialog.
4. In the Add New Project dialog under Project Types, expand the **Other Project Types** node and select **Setup and Deployment**.
5. Under Templates, select **Setup Project**, rename the file and click **OK**.



6. In Solution Explorer, select the Installer project. In the Properties window, select the **ProductName** property and enter the name of your file.

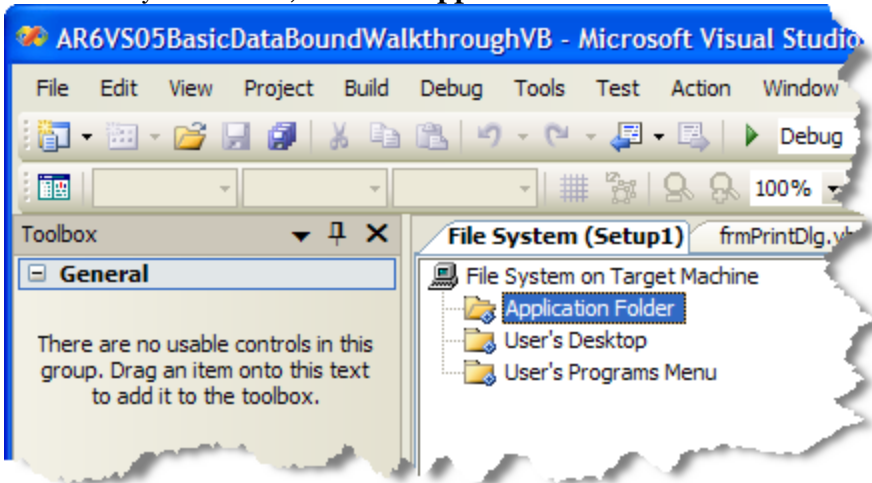
The **ProductName** property determines the name that is displayed for the application in folder names and in the **Add/Remove Programs** dialog box.

To add the ActiveReports application to the installer


1. Select the Installer project in the Solution Explorer. If the File System Editor does not open, drop down the **View**

menu and select **Editor**, then **File System**.

- In the File System Editor, select the **Application Folder**.



- From the Visual Studio **Action** menu, select **Add**, then **Project Output**.
- In the **Add Project Output Group** dialog that appears, choose your ActiveReports project name from the drop-down list.
- In the list, select **Primary Output** and click **OK**. This adds all of the existing assembly dependencies to your project.
- If you want to add other ActiveReports DLLs to the installer (e.g. if you use OleObjects on reports, you need to include the Interop.dll or Interop64.dll for 64-bit machines), in the Solution Explorer, right-click the installer project name, select **Add**, then **Assembly**.

 **Note:** If you would rather use the ActiveReports .msm file, please contact powersupport@grapecity.com.

- In the Select Component dialog that appears, select any components that you want to add and click the **OK** button.
- From the Visual Studio **Build** menu, select **Build YourInstallerProjectName** to build your Installer project.

To deploy the installer application

- Select the Installer project in the Solution Explorer.
- From the Visual Studio **Project** menu, click **Install**.
- The Installer application runs and installs the project on your computer. The distributable exe and msi setup files appear in your installer project Debug folder.

Deploy Web Applications (Std Edition)

Follow this guide to deploy ActiveReports Standard Edition Web projects to your Web server. For Web projects using the Professional Edition WebViewer, see **Deploy Web Applications (Pro Edition)**.

To deploy your ActiveReports Web projects, you must have access to the Microsoft .NET Framework version 2.0 or higher and the coordinating version of ASP.NET. You must also have access to Internet Information Services version 5.1 or 6.0, and you need administrative access to the server. For examples of how to create ActiveReports Web projects, see the walkthroughs linked at the bottom of this topic.

To add referenced DLLs to your project

- In the Visual Studio Solution Explorer, if the References node is not showing, click the **Show All Files** button.

2. Expand the **References** node, and select one of the ActiveReports references.
3. In the Properties window, change the **CopyLocal** property to **True**. The corresponding DLL is stored in the Bin folder of your project.
4. Set the **CopyLocal** property to **True** for each ActiveReports reference used in your project.

To install prerequisites on the server

Follow Microsoft's instructions to install each of the following on your Web server:

- The Microsoft .NET Framework version 2.0 or higher
- Internet Information Services (IIS) version 5.1 or 6.0
- ASP.NET version 2.0 or higher (must be the same version as the Framework)

To copy your project to the server

1. Copy the entire directory containing your project to the server.
2. If your project is in a virtual directory on your local machine (i.e. C:\Inetpub\wwwroot*YourProject*), you must set up a virtual directory in IIS on the server as well.

To set permissions on the server

Depending on your project, you may need to set permissions to allow ActiveReports access to data or folders.

Some examples of required permissions on the server

- If you are saving files (i.e. PDF or RDF) to a folder on Windows XP or 2000 machines, the **ASPNET** user ID needs **Write** access to that folder.
- Windows 2003 is user configurable, so use **the name assigned to the ASPNET user** instead.
- If your application reads anything from any folder, assign **Read** access to it.
- If your reports run on any networked data source (i.e. SQL, Access, etc.) assign **Read** access to it.
- If you use **CacheToDisk**, assign **IsolatedStorageFilePermission** to it.


Localize the End User Report Designer

To localize the designer control

1. In Windows Explorer, navigate to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, navigate to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**).
2. Edit the ARDesigner.bat file:
 - a. Right-click the **ARDesigner.bat** file and select **Edit**.
 - b. Change the culture in the line **set Culture="en-US"** to the culture you want to use. For your convenience, here is a list of predefined .NET **Cultures**.
 - c. Ensure that the **ProgramFilesDDPath** is correct.

 **Caution:** Do not change the **ProjectName**, **dllName**, **msDir**, or **BaseNamespace**.

- d. Save and close the ARDesigner.bat file.
3. Change strings in the resource files:
 - a. Double-click the **ARDesigner.zip** file to open it.
 - b. Extract all of the files to **C:\Program Files\GrapeCity\ActiveReports 6\Localization** (on a **64-bit Windows operating system**, extract the files to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization**). An **ARDesigner** subfolder is created.

 **Note:** If you are working on a 64-bit Windows operating system, you should extract files to C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization.

- c. In the new ARDesigner folder's **Res** subfolder, open each of the resources.resx file and change the strings as needed.
 - d. Drill down in each of the following subfolders and edit the *.resx files as needed: **Designers, Dialogs, ReportExplorer, and ScriptEditor.**
 - e. If you want to change any of the images, rename your localized images to the names of the ones in the **Res\Resources** subfolder and replace them with your localized images.
4. Back in the main Localization folder, double-click the **ARDesigner.bat** file to run it. The NameCompleter.exe application runs, and creates:
 - A SatelliteAssembly folder inside the ARDesigner folder.
 - A language subfolder with the same name as the culture you set in the ARDesigner.bat file inside the SatelliteAssembly folder.
 - A localized ActiveReports.ARDesigner.resources.dll file inside the language subfolder.
 5. Copy the language subfolder and paste it into the Debug folder of your application.

 **Note:** If you want to put your localization in the Global Assembly Cache (GAC), you must first send the localized ActiveReports.ARDesigner.resources.dll file to [GrapeCity](#) and get it signed. Then you can drag the language subfolder with the signed dll file into C:\WINDOWS\ASSEMBLY.

To test your localized application on a machine that does not share the culture of the localized dll

1. Add the following code in the form's constructor just before the InitializeComponent method is called.
2. Replace the "ja" in the example code with the culture specified in the ARDesigner.bat file.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = New
System.Globalization.CultureInfo("ja")
```


To write the code in C#

C# code. Paste **INSIDE** the form's constructor just before the InitializeComponent method.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("ja");
```

Deploy the End User Report Designer (Pro Edition)

To deploy a solution that includes the Designer control (Professional Edition only), you must include the version of the Microsoft Rich Text Edit Control DLL that is installed with ActiveReports in a location like this: **C:\Program Files\Common Files\GrapeCity\ActiveReports 6\riched20.dll** (on a **64-bit Windows operating system**, a location is like this: **C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6\riched20.dll**).

 **Caution:** If you do not deploy Product version 4 (File version 5.40.11.2210) or higher of riched20.dll with the Designer, the user might have a version that is unable to render RTF tables correctly in edit mode of the RichText

control.

Place this file in the same directory as the ActiveReports assemblies.

To deploy riched20.dll

1. Open the Registry Editor.
2. Expand the tree view to My Computer\HKEY_CURRENT_USER\Software\GrapeCity\ActiveReports 6.
3. Right-click in the pane to the right and select **New**, then **String Value**.
4. Name the new String Value **RtfPath**.
5. Double-click **RtfPath**, and enter the path to the newer version of riched20.dll.

Once you have finished these steps, you can deploy your Designer application like any other **Windows Application**.

Customize the FlashViewer Toolbar (Pro Edition)

When you select the FlashViewer ViewerType of the WebViewer (Professional Edition license), the FlashViewer toolbar is very similar to the Viewer control's toolbar. You can show or hide it, reorder buttons, remove buttons, add custom buttons, or create a custom toolbar. Use the Web.Controls namespace to create custom buttons or a custom toolbar that you can specify in the WebViewer's FlashViewerToolbar property.

Hide the Toolbar

Reorder Buttons

Remove a Button


Create a Custom Button

Create a Custom Toolbar

The buttons that are available in the toolbar by default are:

- TOCButton
- PrintButton
- PageRangeButton
- SearchButton
- ZoomOutButton
- ZoomBox
- ZoomInButton
- SinglePageViewButton
- MultiPageBox
- ContinuousViewButton
- PreviousPageButton
- NextPageButton
- CurPageTextArea
- BackwardButton
- ForwardButton

To hide the toolbar

 **Note:** If the **ViewerType** property of your WebViewer control is not set to **FlashViewer**, this code is ignored.

1. In the Visual Studio Solution Explorer, right-click the ASPX file that contains your WebViewer and select **View Designer**.
2. In the design view of your web form, double-click the WebViewer. This creates an event handling method for the Page Load event and takes you to the code view of the page.
3. Use code like the following to hide the toolbar:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Page Load event.

```
WebViewer1.FlashViewerToolBar.Visible = False
```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```
WebViewer1.FlashViewerToolBar.Visible = false;
```

To rearrange buttons in the toolbar

1. In the Visual Studio Solution Explorer, right-click the ASPX file that contains your WebViewer and select **View Designer**.
2. In the design view of your web form, double-click the page. This creates an event handling method for the Page Load event and takes you to the code view of the page.
3. Use code like the following to create a button and insert it at the beginning of the toolbar:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Page Load event.

```
'Get an existing tool from the toolbar. (If you prefer, you can specify the index of the tool.)
Dim tool As DataDynamics.ActiveReports.Web.Controls.ToolBase = WebViewer1.FlashViewerToolBar.Tools("PageRangeButton")
'Remove the tool from the toolbar.
WebViewer1.FlashViewerToolBar.Tools.Remove(tool)
'Add the tool in a different position.
WebViewer1.FlashViewerToolBar.Tools.Insert(0, tool)
```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```
//Get an existing tool from the toolbar. (If you prefer, you can specify the index of the tool.)
ToolBase tool = WebViewer1.FlashViewerToolBar.Tools[ToolsCollection.ToolCommands.PageRangeButton];
```

```
//Remove the tool from the toolbar.
WebViewer1.FlashViewerToolBar.Tools.Remove(tool);
//Add the tool in a different position.
WebViewer1.FlashViewerToolBar.Tools.Insert(8, tool);
```

To remove a button from the toolbar

1. In the Visual Studio Solution Explorer, right-click the ASPX file that contains your WebViewer and select **View Designer**.
2. In the design view of your web form, double-click the WebViewer. This creates an event handling method for the Page Load event and takes you to the code view of the page.
3. Use code like the following to remove a button from the toolbar:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Page Load event.


```
'Get an existing tool from the toolbar. (If you prefer, you can specify the index of the tool.)
Dim tool As DataDynamics.ActiveReports.Web.Controls.ToolBase = WebViewer1.FlashViewerToolBar.Tools("PageRangeButton")
'Remove the tool from the toolbar.
WebViewer1.FlashViewerToolBar.Tools.Remove(tool)
```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```
//Get an existing tool from the toolbar. (If you prefer, you can specify the index of the tool.)
ToolBase tool = WebViewer1.FlashViewerToolBar.Tools[ToolsCollection.ToolCommands.PageRangeButton];
//Remove the tool from the toolbar.
WebViewer1.FlashViewerToolBar.Tools.Remove(tool);
```

To create a custom button and add it to the toolbar

 **Tip:** The ToolsCollection class in the Web.Controls namespace has the standard System.Collections.ObjectModel.Collection methods available, so if you want to just add the button to the end of the toolbar, you can use the **Add** method instead.

1. In the Visual Studio Solution Explorer, right-click the ASPX file that contains your WebViewer and select **View Designer**.
2. In the design view of your web form, double-click the WebViewer. This creates an event handling method for the Page Load event and takes you to the code view of the page.
3. Use code like the following to create a button and insert it at the beginning of the toolbar:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Page Load event.

```
Dim customButton As ToolButton = Tool.CreateButton("CustomButton")
customButton.Caption = "Visit Us!"
customButton.ToolTip = "Click here to visit datadynamics.com"
customButton.ClickNavigateTo = "http://www.datadynamics.com"
'Insert the button at the specified index, in this case 20
'to put it in the second-to-last place, between Backward and Forward.
'Set the index parameter to 0 to put it in the left-most position.
WebViewer.FlashViewerToolBar.Tools.Insert(20, customButton)
```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```
ToolButton customButton = Tool.CreateButton("CustomButton");
customButton.Caption = "Visit Us!";
customButton.ToolTip = "Click here to visit datadynamics.com";
customButton.ClickNavigateTo = "http://www.datadynamics.com";
//Insert the button at the specified index, in this case, 20
//to put it in the second-to-last place, between Backward and Forward.
//Set the index parameter to 0 to put it in the left-most position.
WebViewer.FlashViewerToolBar.Tools.Insert(20, customButton);
```

To create a custom toolbar and add it to the viewer

1. In the Visual Studio Solution Explorer, right-click the ASPX file that contains your WebViewer and select **View Designer**.
2. In the design view of your web form, double-click the WebViewer. This creates an event handling method for the Page Load event and takes you to the code view of the page.
3. Use code like the following to create a custom toolbar and add it to the viewer:

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Page Load event.

```
'Get the collection of buttons and separators used in the toolbar
Dim collection As DataDynamics.ActiveReports.Web.Controls.ToolsCollection = WebViewer1.FlashViewerToolBar.Tools
'Remove all buttons and separators
collection.Clear()
'Add pre-defined buttons
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomOutButton))
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomBox))
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomInButton))
'Add separator
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateSeparator())
'Add pre-defined button
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.SearchButton))
'Add separator
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateSeparator())
'Add custom buttons
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateButton("btn1"))
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateButton("btn2"))
```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```
//Get the collection of buttons and separators used in the toolbar
DataDynamics.ActiveReports.Web.Controls.ToolsCollection collection = WebViewer1.FlashViewerToolBar.Tools;
//Remove all buttons and separators
collection.Clear();
//Add pre-defined buttons
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomOutButton));
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomBox));
```

```
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.ZoomInButton));
//Add separator
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateSeparator());
//Add pre-defined button
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.Create(DataDynamics.ActiveReports.Web.Controls.ToolsCollection.ToolCommands.SearchButton));
//Add separator
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateSeparator());
//Add custom buttons
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateButton("btn1"));
collection.Add(DataDynamics.ActiveReports.Web.Controls.Tool.CreateButton("btn2"));
```

In the **Class Library (on-line documentation)** portion of the documentation, you can see all of the available properties and methods in the Web assembly's Controls namespace.

Localize the Flash Viewer

The FlashViewer, one of the ViewerTypes of the WebViewer control, is localized separately from other ActiveReports resources. You can redistribute the Flash localization resources separately from the application so that you need not recompile the ActiveReports.FlashViewer.swf file.

The default locale is en_US, U.S. English, but the included ActiveReports.FlashViewer.Resources.swf also contains strings localized for ru_RU, Russian, and ja_JP, Japanese.

To localize the FlashViewer for Russian or Japanese

1. Copy the **ActiveReports.FlashViewer.Resources.swf** from **C:\Program Files\GrapeCity\ActiveReports 6\Deployment** (on a **64-bit Windows operating system**, from **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment**) into your project folder that contains the ASPX file with the WebViewer.
2. With focus on the WebViewer control, in the Visual Studio Properties window, expand the **FlashViewerOptions** node.
3. In the **ResourceLocale** property, drop down the list of values and select **ja_JP** or **ru_RU**.
4. Run the project to see the localized FlashViewer.

To create custom localizations

1. In the **C:\Program Files\GrapeCity\ActiveReports 6\Localization** folder (on a **64-bit Windows operating system**, in the **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization** folder), open the **FlashViewer.zip** file.
2. Using Notepad, open the **Resources.properties** file and localize the strings.
3. Email the localized FlashViewer.zip file to powersupport@grapecity.com to have it compiled in the ActiveReports.FlashViewer.Resources.swf file.
4. When you receive the new ActiveReports.FlashViewer.Resources.swf file, place it in your project folder.
5. Open the project in Visual Studio, select the WebViewer, and in the Properties window, expand the **FlashViewerOptions** node.
6. Set the **ResourceLocale** property to the new **culture**.



Note: If you are willing to share your new localization with other customers, let support know so that the updated ActiveReports.FlashViewer.Resources.swf file can be included in future builds of the product.

Deploy Web Applications (Pro Edition)

With ActiveReports 6 Professional Edition, you can set up Web applications for deployment by including the ActiveReports deployment .msm file in your Visual Studio deployment project.

In order to successfully deploy an ActiveReports Web project, the server must have the following installed.

- Microsoft .NET Framework Version 2.0 or higher
- IIS (Internet Information Services) Version 6.0 or higher
- ASP.NET Version 2.0 or higher (must be the same version as the Framework)

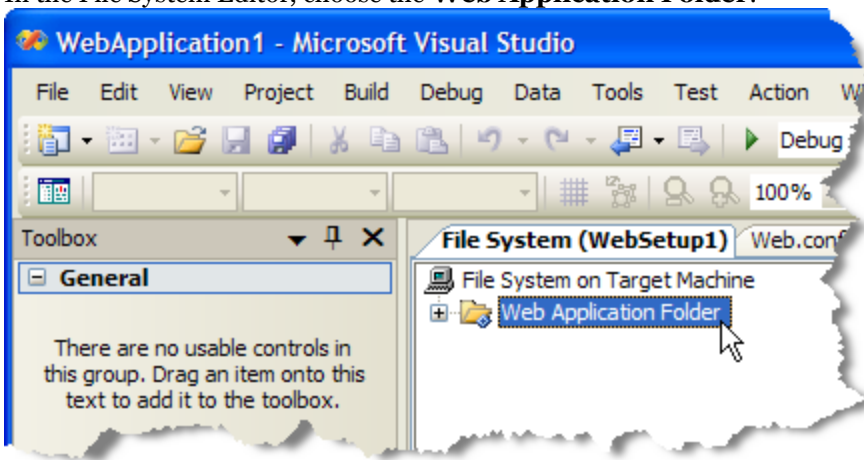
To add an installer project to an ActiveReports project

1. Open an existing ActiveReports Web project or create a new one.
2. From the **Build** menu, select **Build Solution** to build your report project.
3. From the **File** menu, select **Add**, then **New Project**.
4. In the Add New Project dialog that appears, in the Project types pane, expand the **Other Project Types** node and select **Setup and Deployment**.
5. In the Templates pane, select **Web Setup Project**, rename the file and click **OK**. A **File System (YourSetupProjectName)** tab appears.
6. In the Properties window, with **YourSetupProjectName** selected, enter the name of your file in the **ProductName** property.

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

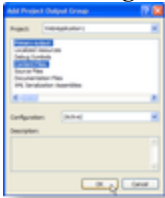
To add the ActiveReports application to the installer

1. Select the installer project in Solution Explorer.
2. In the File System Editor, choose the **Web Application Folder**.



Tip: If the File System Editor is not open, drop down the **View** menu and select **Editor**, then **File System**.

3. From the Visual Studio **Action** menu, select **Add**, then **Project Output**.
4. In the Add Project Output Group window that appears, next to **Project**, select your ActiveReports project name from the drop-down list.
5. Hold down the **Ctrl** key and select **Primary Output** and **Content Files** from the list and click **OK** to add all of the existing assembly dependencies to your Web application.



6. On the **Build** menu, select **Build YourInstallerProjectName** to build your installer project.

Note: If you prefer to use the ActiveReports .msm file, please contact powersupport@grapecity.com.

To deploy the installer application to a Web server

1. In Solution Explorer, select the installer project.
2. From the **Project** menu, select **Install** and follow the wizard's steps to install it.
3. To access the Web application that was deployed, start Internet Explorer and enter the URL:
http://localhost:nnnn where **nnnn** is the port number.



Important: If you are using the WebViewer control or HttpHandlers in your application, you must also **Configure HTTPHandlers (Pro Edition)**.

If you are using the FlashViewer ViewerType of the WebViewer control, you must add the **ActiveReports.FlashViewer.swf** file to your IIS root folder. You can find the file in the C:\Program Files\GrapeCity\ActiveReports 6\Deployment folder (on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment).

If you are using a localized version of the FlashViewer, you also need to add the **ActiveReports.FlashViewer.Resources.swf** file. For more information, see **Localize the Flash Viewer**.

Customize End User Designer Help (Pro Edition)

A scaled-down version of this user guide project is now available for download.

http://downloads.datadynamics.com/ActiveReports6_UserGuide.zip

You can use [Innovasys HelpStudio 3](#) to customize the help to your application, or you can use Adobe Acrobat to customize the included PDF build of the user guide. If you want to distribute the included CHM or PDF file as is for a quick solution, see **Deploy End User Designer Help (Pro Edition)**.

To customize the PDF file (requires [Adobe Acrobat](#))

1. Download the file at: http://downloads.datadynamics.com/ActiveReports6_UserGuide.zip
2. Open the **zip** file, and extract the **AR6Designer.pdf** file.
3. Make any required changes in the file, and save it.
4. Deploy it along with your application.

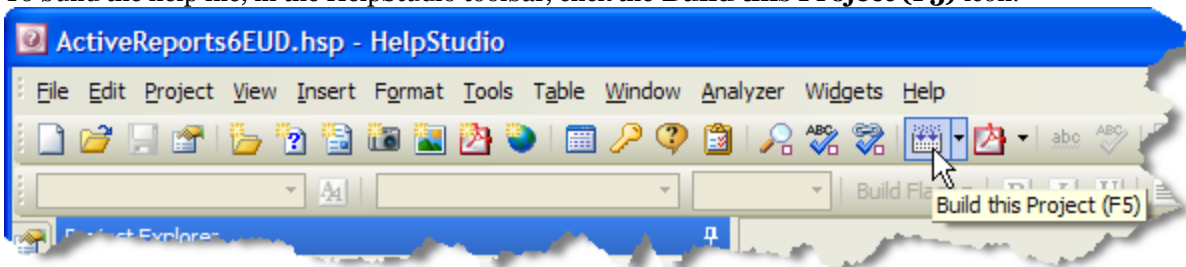
To customize the CHM file (requires [Innovasys HelpStudio 3](#))

For more detailed help on using HelpStudio, see the HelpStudio help file. These steps are just enough to get you started.

1. Download the file at: http://downloads.datadynamics.com/ActiveReports6_UserGuide.zip
2. Open the **zip** file, and extract the contents. The main folder that the project extracts to is **AR6EUD**.
3. In the AR6EUD folder, double-click the **ActiveReports6EUD.hsp** file to open the project in HelpStudio.
4. In the table of contents for the help project, you will notice that some of the topics have a purple flag to the left of the topic name. These topics require editing (mainly converting code samples to script) and are not included in the help file output unless you clear the build flags from them.
 - a. To clear the build flag from a topic, right-click the topic in the table of contents and select **Topic Properties**.
 - b. In the Topic Properties dialog, under the Build Flags node, clear the check box next to **In Progress**.
 - c. Click the **OK** button to close the dialog and save the change.
5. Double-click any topic in the table of contents to open it for editing. You will also notice that content in some of the overview topics is highlighted with the purple In Progress build flag.
 - a. To clear the build flag from content, select the content and in the toolbar, drop down the **Build Flags** list.
 - b. Select **Remove Build Flag**. The selected content no longer appears highlighted in purple, and will be

included in the help file output.

6. If you want to find and replace specific text within the entire project, for example "ActiveReports" or "GrapeCity," from the **Tools** menu, select **Project Find and Replace**.
 - a. Enter the text you want to find in the **Find What** field, and the text you want to replace it with in the **Replace With** field.
 - b. Click the **Find in Project** button or hit **Enter** on your keyboard. The list of topics containing the search terms appears.
 - c. Double-click any item in the list to open the topic for editing. The find dialog appears and the search term is highlighted in the topic.
7. If you add or remove any topics and you are providing context-sensitive help, you need to regenerate the **ActiveReports6.h** file.
 - a. From the **Tools** menu, select **Create .h Help Context ID File**.
 - b. In the Create .h File dialog that appears, click the **Browse** button.
 - c. In the Browse dialog that appears, select the **ActiveReports6.h** file and click **OK**.
 - d. Select the check box to **Assign Help Context IDs to any Topics without IDs already assigned**.
 - e. Change the **#define Prefix** to **AR6_** or to the value you want to use, and click the **OK** button to create the file.
8. To change the name and other properties of the generated help file, in the Project Explorer, expand the **Build Profiles** node.
 - a. Double-click the **AR6Designer (Compiled HTML Help 1.x file)** build profile to open the Build Profile dialog.
 - b. Make any changes to the build profile and click the **OK** button.
9. To change the name and other properties of the generated PDF booklet, in the Project Explorer, expand the **Booklets** node.
 - a. Double-click the **AR6Designer** booklet to open the booklet properties tab.
 - b. Make any changes to the booklet and save the project.
10. To build the help file, in the HelpStudio toolbar, click the **Build this Project (F5)** icon.



11. In the Build Options dialog that appears, select the check boxes that you want to build and click the **Build** button. The compiled CHM or PDF files appear in the Build folder.

Deploy End User Designer Help (Pro Edition)

A scaled-down version of this user guide project is now available for download.

http://downloads.datadynamics.com/ActiveReports6_UserGuide.zip

The PDF version of the file is also available.

<http://downloads.datadynamics.com/AR6DesignerGuide.pdf>

You can use [Innovasys HelpStudio 3](#) to customize the help to your application, or you can use [Adobe Acrobat](#) to customize the included PDF build of the user guide. Or you can distribute the included CHM or PDF file as is for a quick solution. For information on customizing the user guide, see **Customize End User Designer Help (Pro Edition)**.

To deploy end user designer help

1. Download the file at: http://downloads.datadynamics.com/ActiveReports6_UserGuide.zip
2. Open the **zip** file. In it you will find all of the help project files, as well as the **AR6Designer.pdf** and **ActiveReports6.chm** deliverables and the **ActiveReports6.h** file.
3. The **ActiveReports6.h** file contains context IDs mapped to all of the topics in the user guide. You can use this to provide context-sensitive help for your users.
 - a. If you want to provide context-sensitive help, open **ActiveReports6.h** in notepad.
 - b. Find the topic names that you want to associate with your application contexts and enter a context ID from the .h file into your application UI element's **Help Context ID** property for each topic.
 - c. Build your application. When you run the application and give focus to one of the UI elements for which you specified a Help Context ID, clicking the F1 key opens the associated help topic.
4. Include the **ActiveReports6.chm** and **ActiveReports6.h** files in your deployment folder along with your executable.
5. You can put the **AR6Designer.pdf** file in any folder for deployment.

Configure HTTPHandlers (Pro Edition)

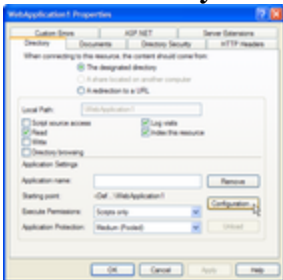
Manually Configuring Web Samples

In order to use ActiveReports HTTPHandlers on the Web with ASP.NET, you must first configure the machine to use the handlers. Please note that HttpHandlers are only enabled if you purchased the Professional Edition.

For information on how to configure ActiveReports handler mappings in IIS 7.x, see the section **Configure Handler Mappings in IIS 7.x** of this Guide.

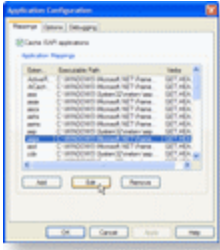
To configure the report layout (RPX) handler

1. In the Control Panel, open **Administrative Tools**, then **Internet Information Services**.
2. In the Internet Information Services window that appears, expand the tree view in the left pane until you see the Web site that you need to configure. Right-click the Web site and select **Properties**.
3. On the **Directory** tab of the *YourWebSite* Properties dialog that appears, click the **Configuration** button.

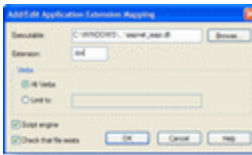


4. In the Application Configuration dialog that appears, select the list item with **.aspx** in the Extension column and click the **Edit** button.

Note: If your machine does not have the ASP.NET server components installed, the .aspx handler does not appear in the Application Mappings list.



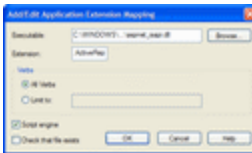
5. In the **Executable** field, select and copy all of the text, and click **Cancel** to return to the Application Configuration dialog.
6. Click the **Add** button to add a new Application Mapping.
7. In the Add/Edit Application Extension Mapping window that appears, paste the value from the .aspx extension into the **Executable** field.



8. In the **Extension** field, enter **.rpx**.
9. Select the "Check that file exists" checkbox.
10. Click the **OK** button to add the mapping and return to the Application Configuration dialog.

To configure the compiled report handler

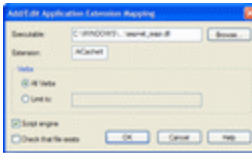
1. In the Application Configuration dialog, click the **Add** button to add a new Application Mapping.
2. In the **Executable** field, paste the value copied from .aspx above.
3. In the **Extension** field, enter **.ActiveReport**.
4. Clear the **Check that file exists** check box.



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

To configure the WebCacheAccessHandler

1. In the Application Configuration dialog, click the **Add** button to add a new Application Mapping.
2. In the **Executable** field, paste the value copied from .aspx above.
3. In the **Extension** field, enter **.ArCacheItem**.
4. Clear the **Check that file exists** check box.



5. Click the **OK** button 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.

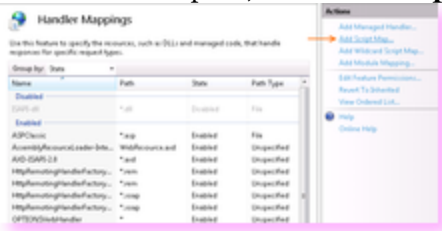
See the topic **Add Report Links to Web Forms (Pro Edition)** for information on enabling the handlers in a Web Form.

Configure Handler Mappings in IIS 7.x

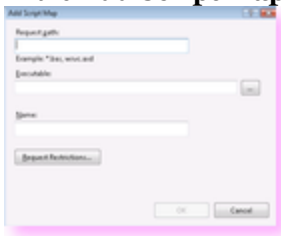
Follow these steps to configure the ActiveReports handler mappings in IIS 7.x for running ActiveReports Web Applications on your machine.

To configure the ActiveReports handler mappings in IIS 7.x to run ActiveReports Web Applications on your machine (classicMode)

1. In the Control Panel, open **Administrative Tools**, then **Internet Information Services Manager**.
2. In the **Internet Information Services** window that appears, expand the tree view in the left pane until you see the Web application you need to configure.
3. Select the node for your application. The **Features View** pane is displayed.
4. Double-click **Handler Mappings** in the **Features View** pane.
5. On the **Actions** pane, click **Add Script Map...**



6. In the **Add Script Map** dialog box that appears, enter the following information:



Request path: *.ActiveReport

Executable: To choose the executable, depending on your version of the .NET Framework, see the table below.

Version of .NET Framework

Executable

| | |
|------------------------------|--|
| ASP.NET 2.0 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 2.0 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 4 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll |
| ASP.NET 4 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll |

Name: ActiveReport Script Mapping

Note: If you have a 64 bit machine, you need to add script mappings for the 64 bit version for aspnet_isapi.dll (C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll) in the same way as it has been done for the 32 bit version in the web.config file.

7. Click the **Request Restrictions...** button and make sure the **"Invoke handler only if request is mapped to:"** check box is not selected.

8. Click **OK** to close the **Add Script Map** window.
9. Repeat steps 5-8 to add another script mapping.

Enter the following information for the second script mapping (see the step 6 above):

Request path: *.ArCacheItem

Executable: To choose the executable, depending on your version of the .NET Framework, see the table below.

Version of .NET Framework

Executable

| | |
|------------------------------|--|
| ASP.NET 2.0 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 2.0 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 4 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll |
| ASP.NET 4 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll |

Name: ActiveReport Cache Item Script Mapping



Note: If you have a 64 bit machine, you need to add script mappings for the 64 bit version for aspnet_isapi.dll (C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll) in the same way as it has been done for the 32 bit version in the web.config file.

10. Repeat steps 5-8 to add the last required script mapping. For the **step 7 above**, make sure the **"Invoke handler only if request is mapped to:"** and the **"File"** check boxes are selected.

Enter the following information for the third script mapping (see the step 6 above):

Request path: *.rpx

Executable: To choose the executable, depending on your version of the .NET Framework, see the table below.

Version of .NET Framework

Executable

| | |
|------------------------------|--|
| ASP.NET 2.0 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 2.0 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll |
| ASP.NET 4 (32 bit version) | C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll |
| ASP.NET 4 (64 bit version) | C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll |

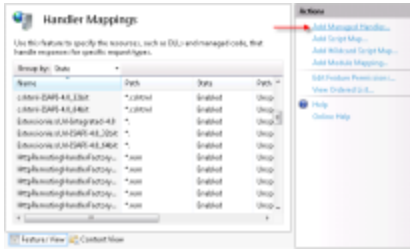
Name: ActiveReport RPX Script Mapping



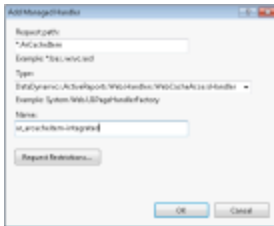
Note: If you have a 64 bit machine, you need to add script mappings for the 64 bit version for aspnet_isapi.dll (C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll) in the same way as it has been done for the 32 bit version in the web.config file.

To configure the ActiveReports handler mappings in IIS 7.x to run ActiveReports Web Applications on your machine (integratedMode)

1. In the Control Panel, open **Administrative Tools**, then **Internet Information Services Manager**.
2. In the **Internet Information Services** window that appears, expand the tree view in the left pane until you see the Web application you need to configure.
3. Select the node for your application. The **Features View** pane is displayed.
4. Double-click **Handler Mappings** in the **Features View** pane.
5. On the **Actions** pane, click **Add Managed Handler...**



6. In the **Add Managed Handler** dialog box that appears, enter the following information:



Request Path: *.ArCacheItem

Type: DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler

Name: ar_arcacheitem-integrated

7. Click the **Request Restrictions...** button and make sure the **"Invoke handler only if request is mapped to:"** check box is not selected.
8. Click **OK** to close the **Add Managed Handler** window.
9. Repeat steps 5-8 to add another script mapping.

Enter the following information for the second script mapping (see the step 6 above):

Request path: *.ActiveReport

Type: DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler

Name: ar_activereport-integrated

10. Repeat steps 5-8 to add another script mapping. For the **step 7 above**, make sure the **"Invoke handler only if request is mapped to:"** and the **"File"** check boxes are selected.



Enter the following information for the third script mapping (see the step 6 above):

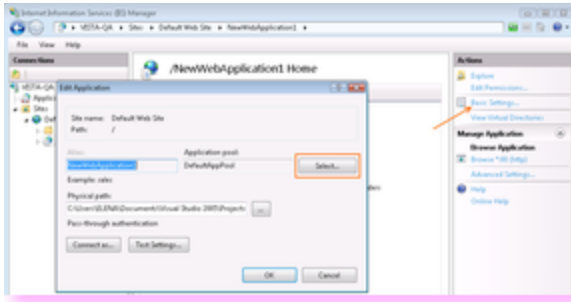
Request path: *.rpx

Type: DataDynamics.ActiveReports.Web.Handlers.RpxHandler

Name: ar_rpx-integrated

To check whether your Web Application is running in the Classic .NET Application Pool

1. In the **Internet Information Services Manager**, select your Web Application in the **Connections** panel.
2. In the **Actions** pane, click **Basic Settings...** The **Edit Application** window will appear.



3. In the **Edit Application** window, click the **"Select..."** button.
4. In the drop-down box, select **Classic .NET AppPool**, then click **OK**.



5. Click **OK** in the **Edit Application** window to accept the changes.

To add handlers without configuring IIS 7.x when your Application pool is Classic .NET AppPool (classicMode)

In the <system.web> tag, paste the following code.

```
<httpHandlers>
  <add verb="*" path="*.rpx"
type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
  <add verb="*" path="*.ActiveReport"
type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
  <add verb="*" path="*.ArCacheItem"
type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
</httpHandlers>
```

IMPORTANT: Change the Version number in the pasted code to match the version installed on your machine.


In <system.webServer> tag, paste the following code.

```
<handlers>
  <add name="ar rpx" path="*.rpx" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv2.0,bitness32" />
  <add name="ar cache" path="*.ArCacheItem" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv2.0,bitness32" />
  <add name="AR" path="*.ActiveReport" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode,runtimeVersionv2.0,bitness32" />
</handlers>
```

Note: If you have a 64 bit machine, you need to update the code in all three handlers above as follows:

1. Change the scriptProcessor path for the aspnet_isapi.dll to


- "C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll".
2. Change the preCondition attribute to "classicMode, runtimeVersionv2.0, **bitness64**".
 3. In ASP.NET 4 (ClassicMode), change the preCondition attribute to "classicMode, runtimeVersion4.0, bitness64".

 **IMPORTANT:** Ensure that the .NET Framework Version number in the pasted code matches the version installed on your machine.


To add handlers without configuring IIS 7.x when your Application pool is DefaultAppPool (integratedMode)

In the <system.web> tag, paste the following code.

```
<handlers>
  <add name="*.ArCacheItem_*" path="*.ArCacheItem" verb="*"
type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff"
preCondition="integratedMode, runtimeVersionv2.0" />
  <add name="*.ActiveReport_*" path="*.ActiveReport" verb="*"
type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff"
preCondition="integratedMode, runtimeVersionv2.0" />
  <add name="*.rpx_*" path="*.rpx" verb="*"
type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
Version=6.0.2019.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff"
preCondition="integratedMode, runtimeVersionv2.0" />
  <add name="ar rpx" path="*.rpx" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode, runtimeVersionv2.0, bitness32" />
  <add name="ar cache" path="*.ArCacheItem" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode, runtimeVersionv2.0, bitness32" />
  <add name="AR" path="*.ActiveReport" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode, runtimeVersionv2.0, bitness32" />
</handlers>
```

 **Note:** If you have a 64 bit machine, you need to update the code in all three handlers above as follows:

1. Change the scriptProcessor path for the aspnet_isapi.dll to "C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll".
2. Change the preCondition attribute to "classicMode, runtimeVersionv2.0, **bitness64**".
3. In ASP.NET 4 (ClassicMode), change the preCondition attribute to "classicMode, runtimeVersion4.0, bitness64".

 **IMPORTANT:** Change the Version number in the pasted code to match the version installed on your machine.

In the <system.webServer> tag, paste the following code.

```
<handlers>
  <add name="ar rpx" path="*.rpx" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" preCondition="classicMode, runtimeVersionv2.0, bitness32" />
  <add name="ar cache" path="*.ArCacheItem" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
```

```
resourceType="Unspecified" precondition="classicMode, runtimeVersionv2.0, bitness32" />
  <add name="AR" path="*.ActiveReport" verb="*" modules="IsapiModule"
scriptProcessor="C:\Windows\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll"
resourceType="Unspecified" precondition="classicMode, runtimeVersionv2.0, bitness32" />
</handlers>
```

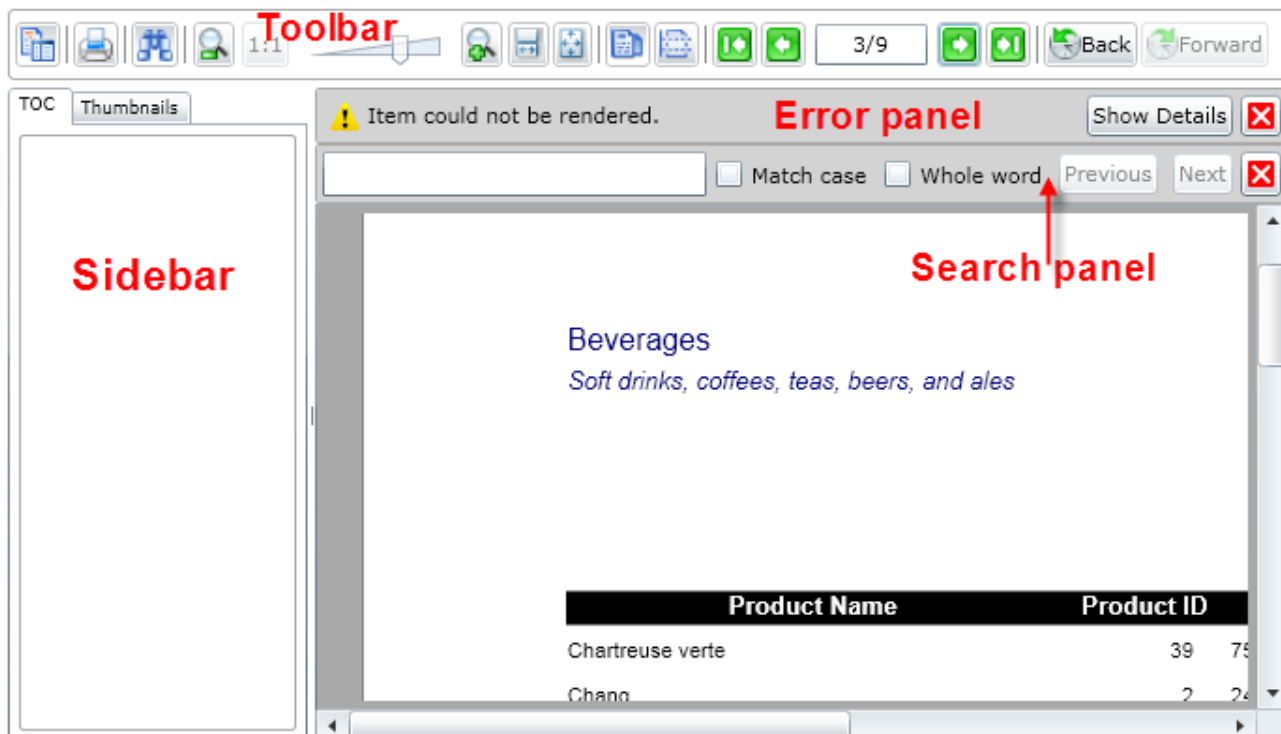
Note: If you have a 64 bit machine, you need to update the code in all three handlers above as follows:

1. Change the scriptProcessor path for the aspnet_isapi.dll to "C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_isapi.dll".
2. Change the precondition attribute to "classicMode, runtimeVersionv2.0, **bitness64**".
3. In ASP.NET 4 (ClassicMode), change the precondition attribute to "classicMode, runtimeVersion4.0, bitness64".

IMPORTANT: Ensure that the .NET Framework Version number in the pasted code matches the version installed on your machine.

Customize the Silverlight Viewer (Pro Edition)

The ActiveReports Silverlight Viewer is a customizable control. You can easily change the look of the Silverlight Viewer and such its elements as the **error panel**, **search panel**, **sidebar** and **toolbar** by modifying properties in the default Silverlight Viewer template (DefaultSLViewerTemplates.xaml).



To add the customization template to the Silverlight project

1. Open your Silverlight project or create a new Silverlight project as described in **Silverlight Viewer (Pro Edition)**.
2. In Solution Explorer, select the *YourProject* directory.
3. On the Visual Studio **Project** menu, click **Add Existing item**.
4. In the dialog that appears, locate and select DefaultSLViewerTemplates.xaml and click **OK**. You can find DefaultSLViewerTemplates.xaml in the C:\Program Files\GrapeCity\ActiveReports6\Deployment\Silverlight\Templates

folder (on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment\Silverlight\Templates).

5. On MainPage.xaml, add the following code to the <UserControl> section:

Paste to the UserControl section on Design view of MainPage.xaml

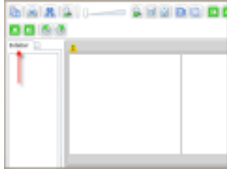
```
<UserControl.Resources>
    <ResourceDictionary Source="DefaultSLViewerTemplates.xaml" />
</UserControl.Resources>
```

To customize the Silverlight Viewer sidebar

1. In Solution Explorer, double-click DefaultSLViewerTemplates.xaml.
2. In the file that opens, search for "**sidebar implementation**".
3. In the Header property of <sdk:TabItem>, type "**Sidebar**".

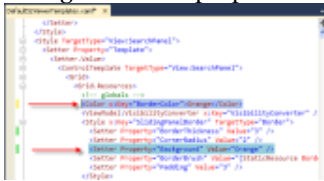


4. Open MainPage.xaml to see the Sidebar caption in the TOC view of the Silverlight Viewer.

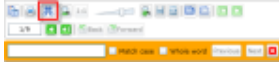


To customize the Silverlight Viewer search panel

1. In Solution Explorer, double-click DefaultSLViewerTemplates.xaml.
2. In the file that opens, search for "**searchpanel**".
3. Change the Color properties in the ControlTemplate section for View:SearchPanel.



4. Press **F5** to see the customized search panel.



To add a customized button to the Silverlight Viewer toolbar

1. In Solution Explorer, select the *YourProjectName* directory.
2. On the Visual Studio Project menu, select **Add Class**.
3. In the **Add New Item** dialog that appears, select **Class**, rename it to **MyCommand** and click **Add**.
4. In the MyCommand.cs that opens, add the following code to implement a command:

C# code. Add to AboutUsCommand.cs

```
public class MyCommand : ICommand
{
    public bool CanExecute(object parameter)
    {
        return true;
    }

    public void Execute(object parameter)
    {
        MessageBox.Show("GrapeCity is the world's largest component vendor.", "About Us", MessageBoxButton.OK);
    }
}
```



```

        public event EventHandler CanExecuteChanged;
    }

```

5. In the Solution Explorer, select the *YourProjectName* directory.
6. On the Visual Studio **Project** menu, select **Add Existing Item**.
7. In the **Add Existing Item** that opens, select the DefaultSLViewerTemplates.xaml and click **Add**.
8. In the dialog that appears, locate and select DefaultSLViewerTemplates.xaml and click **OK**. You can find DefaultSLViewerTemplates.xaml in the C:\Program Files\GrapeCity\ActiveReports6\Deployment\Silverlight\Templates folder (on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment\Silverlight\Templates).
9. On MainPage.xaml, add the following code to the <UserControl> section:

Paste to the UserControl section on Design view of MainPage.xaml

```

<UserControl.Resources>
    <ResourceDictionary Source="DefaultSLViewerTemplates.xaml" />
</UserControl.Resources>

```

10. In Solution Explorer, double-click DefaultSLViewerTemplates.xaml.

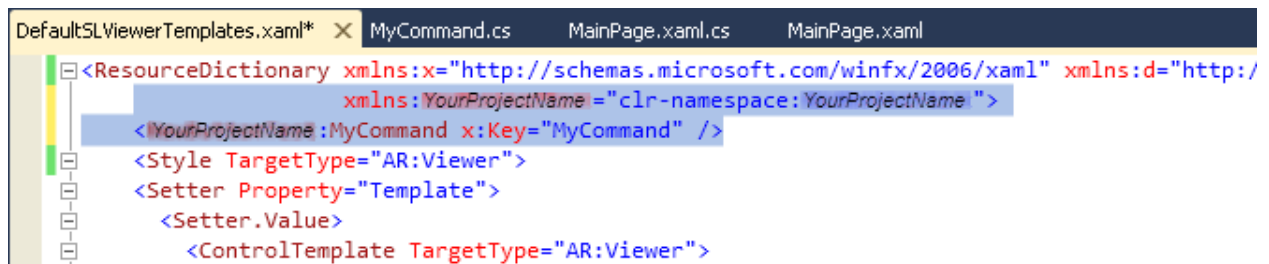
11. In the file that opens, add the following code.

XML code. Add to DefaultSLViewerTemplates.xaml

```

<ResourceDictionary>
    ...
    xmlns:YourProjectName="clr-namespace:YourProjectName">
    <YourProjectName:MyCommand x:Key="MyCommand" />
    ...
</ResourceDictionary>

```



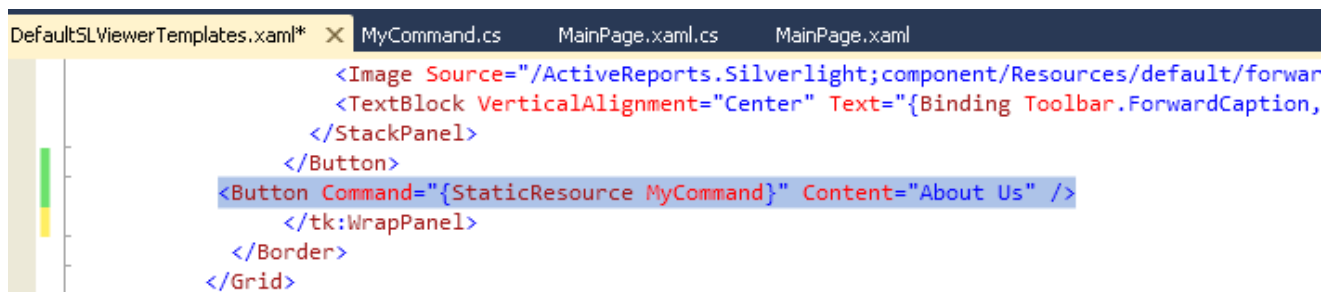
12. In the same file, add the following code to add a button.

XML code. Add to DefaultSLViewerTemplates.xaml inside the Grid tags

```

<Button Command="{StaticResource MyCommand}" Content="About Us" />

```



Note: ActiveReports 6 includes several predefined themes that you can find in the C:\Program Files\GrapeCity\ActiveReports 6\Deployment\Silverlight\Templates folder.

On a **64-bit Windows operating system**, you can find the predefined themes in the C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment\Silverlight\Templates folder.


Localize the Silverlight Viewer (Pro Edition)

The Silverlight Viewer, a control available in Visual Studio 2010 Silverlight projects, is localized separately from other ActiveReports resources.

The default locale is en_US, U.S. English, but the installation includes ActiveReports.Silverlight.resources.dll files with strings localized for ja-JP, Japanese, ru-RU, Russian, and zh-CN, Chinese.

To localize the Silverlight Viewer for Japanese, Russian or Chinese with included resources

1. Copy the required **ja-JP**, **ru-RU** or **zh-CN** folder from **C:\Program Files\Common Files\GrapeCity\ActiveReports 6\ja-JP (or ru-RU or zh-CN)** into the **bin\Debug** folder of the Silverlight project that contains the XAML file with the Silverlight Viewer. On a **64-bit Windows operating system**, this file is located in **C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6\ja-JP (or ru-RU or zh-CN)**.
2. In the Solution Explorer, right-click the project file that you want to localize, and select **Unload Project**. This allows you to edit the file. The file name in Solution Explorer is appended with **(unavailable)**.

 **Tip:** Alternatively, you can edit the **csproj** or **vbproj** file in Notepad outside of Visual Studio.

- a. Right-click the project file and select **Edit MyProject**.
- b. In the **SupportedCultures** node, add one or more locales that you want to use so that it looks like the following.

SupportedCultures. Add locales that you want to use.

```
<SupportedCultures>zh-CN, ja-JP, ru-RU</SupportedCultures>
```

- c. When you have finished, right-click the project file in Solution Explorer, and select **Reload Project**.
3. In the **MainPage.xaml.vb** or **.cs** file, add code like the following to the MainPage class, changing the CultureInfo to the locale that you want to use.
To write the code in Visual Basic

Visual Basic Code. Paste INSIDE the MainPage class.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = New  
System.Globalization.CultureInfo("ja-JP")  
InitializeComponent()
```

To write the code in C#

C# Code. Paste INSIDE the MainPage class.


```
System.Threading.Thread.CurrentThread.CurrentUICulture = new  
System.Globalization.CultureInfo("ja-JP");  
InitializeComponent();
```

4. Run the project to see the localized Silverlight Viewer.

To create custom localizations

1. In the Solution Explorer, right-click the project file that you want to localize, and select **Unload Project**. This

allows you to edit the file. The file name in Solution Explorer is appended with **(unavailable)**.

 **Tip:** Alternatively, you can edit the **csproj** or **vbproj** file in Notepad outside of Visual Studio.

- a. Right-click the project file and select **Edit MyProject**.
- b. In the **SupportedCultures** node, add one or more locales that you want to use so that it looks like the following.

SupportedCultures. Add locales that you want to use.

```
<SupportedCultures>zh-CN, ja-JP, ru-RU</SupportedCultures>
```

- c. When you have finished, right-click the project file in Solution Explorer, and select **Reload Project**.
2. In the **MainPage.xaml.vb** or **.cs** file, add code like the following to the MainPage class, changing the CultureInfo to the locale that you want to use.

To write the code in Visual Basic

Visual Basic Code. Paste INSIDE the MainPage class.

```
System.Threading.Thread.CurrentThread.CurrentUICulture = New
System.Globalization.CultureInfo("ja-JP")
InitializeComponent()
```

To write the code in C#

C# Code. Paste INSIDE the MainPage class.


```
System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("ja-JP");
InitializeComponent();
```


3. In the **C:\Program Files\GrapeCity\ActiveReports 6\Localization** folder (on a **64-bit Windows operating system**, in the **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization** folder), open the **SilverlightViewer.zip** file.
4. Using Notepad, open the **Resources.resx** file and localize the strings contained in the `<value>` `</value>` tags.
5. Save the **Resources.resx** file.
6. In the **C:\Program Files\GrapeCity\ActiveReports 6\Localization** folder (on a **64-bit Windows operating system**, in the **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Localization** folder), edit the **SilverlightViewer.bat** file:
 - a. Right-click the SilverlightViewer.bat file and select Edit.
 - b. Change the culture in the line `set Culture="ja-JP"` to the culture you want to use.
 - c. Save and close the SilverlightViewer.bat file.
7. Back in the main Localization folder, double-click the **SilverlightViewer.bat** file to run it. The language folder **ja-JP** will be created in the SilverlightViewer folder.
8. Email the localized ja-JP.zip file to powersupport@grapecity.com to have the ActiveReports.Silverlight.resources.dll signed.
9. When you receive the new ActiveReports.Silverlight.resources.dll file, open the **ClientBin** folder of your Web Application.
10. In the **ClientBin** folder of your Web Application, rename the extension of the XAP file from **.XAP** to **.zip** and decompress it.
11. Copy the language folder with the signed ActiveReports.Silverlight.dll into the XAP decompressed folder.
12. Open the project in Visual Studio.
13. In Solution Explorer, open the **AppManifest.XAML** file in the *YourProject*\Properties folder and add the `<AssemblyPart>` element to the `<Deployment.Parts>` `</Deployment.Parts>` tags as follows (you can modify the culture string to the one you want):

Paste after <Deployment.Parts> of AppManifest.XAML

```
<AssemblyPart Source="culture/ActiveReports.Silverlight.resources.dll" />
```

14. Save the **AppManifest.XAML** file and compress the XAP folder to the ZIP file.

 **Important:** We recommend that you do NOT build or rebuild the application after you complete the localization steps; otherwise the XAP file will be changed to the original one.

 **Note:** If you are willing to share your new localization with other customers, let support know so that the updated ActiveReports.Silverlight.resources.dll file can be included in future builds of the product.

Samples and Walkthroughs

To understand some of the more complex tasks you can accomplish using ActiveReports, you can open included sample projects, or you can follow walkthroughs, step-by-step tutorials that walk you through every step required to create a specific type of report.

This section contains:

Samples

Browse brief descriptions of included samples, and follow links that open sample projects in Visual Studio.

Walkthroughs

Look through tutorials that teach you all of the steps involved in creating various types of ActiveReports projects, from the basic report through more complex unbound reports and Web options.

Samples

Your ActiveReports 6 installation includes the following samples, with four versions of each. There is a **C#** and a **Visual Basic.NET** version for Visual Studio versions **2005**, **2008** and **2010**.

Click a sample name below to drop down a description of the sample. A more complete description of the samples are available in a ***README.html** file within the sample folder, and comments are sprinkled throughout the sample code.

Annual Report

Demonstrates how to use subreports and nested subreports, how to modify data source properties at run time, how to use parameters in the chart control, how to create alternate row highlighting and how to use the page break control. See the **Annual Report Sample** topic for more information.

Bound Data

Demonstrates binding to ADO.NET Data objects. See the **Bound Data Sample** topic for more information.

Calculated Fields

Demonstrates using an unbound data field to perform a calculation which can then be aggregated.

Category Selection

Demonstrates using the ad hoc report filter by modifying the report's SQL query at run time. See the **Category Selection Sample** topic for more information.

Charting

Demonstrates the use of the Chart control in both bound and unbound modes. See the **Charting Sample** topic for more information.

NEW! Cross Section Controls

Demonstrates the use of the new cross section lines and boxes. See the **NEW Cross Section Control Sample** topic for more information.

Cross Tab Report

Demonstrates using unbound data, conditional highlighting and distributing data across columns to create a cross-tab view and data aggregation. See the **Cross Tab Report Sample** topic for more information.

Custom Preview

Demonstrates using viewer control customization, export filters, rich edit control and mail-merge, parameters in the chart control, and grouping. See the **Custom Preview Sample** topic for more information.

Data Field Expressions

Demonstrates the use of expressions in the DataField properties of controls.

Hyperlinks and Drill Down

Demonstrates using hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another. See the **Hyperlinks and Drill Down Sample** topic for more information.

IList Binding

Demonstrates how to bind reports to objects.

Print Multiple Pages per Sheet

Demonstrates how to print multiple pages of a report on a single sheet of paper.

RDF Viewer

Demonstrates customizing the WinForms viewer control toolbar, loading Report Document Files (RDF) and using the export filters. See the **Rdf Viewer Sample** topic for more information.

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 Flash viewer to view report output on the client machine.

NEW! Style Sheets

Demonstrates the use of style sheets to create consistent styles across different reports. See the **NEW Style Sheets Sample** topic for more information.

Subreports

Shows the proper use of Reports with SubReports to minimize memory usage. See the **SubReports Sample** topic for more information.

Unbound Data

Demonstrates retrieving data from an array and from a text file in unbound mode.

XML

Demonstrates the XML data source and using it to run multi-level reports with and without using subreports.

Professional Edition

NEW! Flash Web Viewer

Demonstrates customization possibilities with the new FlashViewer, including localization, themes, and a custom button. See the **NEW Flash Web Viewer Sample** for more information.

End User Designer

Demonstrates a custom end-user report designer that can be integrated in your applications to allow users to modify report layouts.

Toolbox Class Library

This is the project that creates the toolbox used in the End User Designer sample.

Professional Edition Web Sample

The ASP.NET Web Samples demonstrate the use of Professional Edition ASP.NET features, including HTTP Handlers, Report Caching, and the Server Viewer Control in both Visual Studio 2005 and 2008.



Note: The ASP.NET user machine must have ASP.NET write access before you run the sample or an exception is thrown during execution.

NEW! Silverlight Viewer Sample

Demonstrates the new Silverlight Viewer with its various options of loading RPX and RDF reports. The sample also demonstrates how to print a report in PDF, customize a Viewer theme and localize the Viewer.

The Silverlight Viewer Sample (OOB) demonstrates the out-of-browser support of Silverlight-based applications. See **Silverlight Viewer (Pro Edition)** for more information.

NEW Silverlight Viewer Sample

The Silverlight Viewer sample demonstrates the new Silverlight Viewer with its various options of loading RPX and RDF reports. It also demonstrates how to print a report in PDF, customize a Viewer theme and localize the Viewer.

When you run the project, the Default.aspx page appears in your browser, and you will see a number of choices for options that you can set on the Silverlight Viewer.

The solution contains two projects with the following files and folders:

SilverlightViewerSample Project

- **Images** - a folder that contains the images used on the main page.
- **Resources** - a folder that contains the report used in the sample. You can alter the project and add your own reports to this folder.
- **Templates** - a folder that contains the templates used to change the look and feel of the viewer. You can alter these templates to create your own custom look for the viewer.
- **App.xaml** - a file used by Silverlight applications to declare shared resources.
- **Commands.vb (or .cs)** - a class used to implement the **About** and **Open File** commands when the user opts to add buttons to the viewer.
- **MainPage.xaml** - a user control that contains the Silverlight Viewer. The code behind this file, MainPage.xaml.vb (or .cs), handles the user's choices passed in from the main page.
- **SLVOptions.vb (or .cs)** - a class used to set up public properties passed by the main page.

SilverlightViewerSample.Web Project

- **ClientBin** - a folder that contains static RDF files of reports used in the project, and the XAP file which contains the compressed assemblies and resources for the application.
- **Images** - a folder that contains the images used on the main page.
- **Reports** - a folder that contains the RPX files of reports used in the sample. You can alter the project and add your own reports to this folder.
- **Default.aspx** - a Web Form that shows the introductory text and controls. The code behind this file, Default.aspx.vb (or .cs), handles the button click event that calls the SilverlightViewer.aspx.
- **IRdfService.vb (or .cs)** - an interface used to implement the **GetReport** method when the user opts to load the report from an RDF stream.
- **RdfService.svc** - a service that contains the **GetReport** method.
- **RdfStream.aspx** - a Web Form that handles the various ways that you can load the report from a stream. See the code behind this file, RdfStream.aspx.vb (or .cs), for more information.
- **Silverlight.js** - a helper file for Silverlight installation and instantiation.
- **SilverlightViewer.aspx** - a Web Form that contains the Silverlight viewer.
- **Web.config** - a configuration file that contains the httpHandlers that allows ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

NEW Silverlight Viewer Sample (OOB)

The Silverlight Viewer sample (OOB) demonstrates how to use the new Silverlight Viewer for out-of-browser (OOB) applications. When you run the project, you will see a number of choices for options that you can set on the Silverlight

Viewer.

The solution contains the following files and folders:

SilverlightViewerSample (OOB)

- **Images** - a folder that contains the images used on the main page.
- **Reports** - a folder that contains the report used in the sample. You can alter the project and add your own reports to this folder.
- **Templates** - a folder that contains the templates used to change the look and feel of the viewer. You can alter these templates to create your own custom look for the viewer.
- **App.xaml** - a file used by Silverlight applications to declare shared resources.
- **Commands.vb (or .cs)** - a class used to implement the **About** and **Open File** commands when the user opts to add buttons to the viewer.
- **Introduce.xaml** - a user control that contains the text and controls used in the main page. The code behind this file, Introduce.xaml.vb (or .cs) handles the button click event, calling the ShowSilverlightViewer function in the main page.
- **MainPage.xaml** - a user control that handles whether to show the introductory text and controls or the Silverlight Viewer. Initially, the Silverlight Viewer has its Visibility set to Collapsed (see the XAML code). The code behind this file, MainPage.xaml.vb (or .cs), handles the button click event that controls the toggling between ShowSilverlightViewer and ShowIntroduce, and the GetOptions function that collects the user's choices and passes them to the viewer.
- **SilverlightViewer.xaml** - a user control that contains the Silverlight Viewer. The code behind this file, SilverlightViewer.xaml.vb (or .cs), handles the user's choices passed in from the main page.
- **SLVOptions.vb (or .cs)** - a class used to set up public properties passed by the main page.

NEW Flash Web Viewer Sample

This sample is located at the following location:

[User Documents folder]\GrapeCity\ActiveReports 6\Samples\CSharp\Professional\ArWebSampleProCs6.

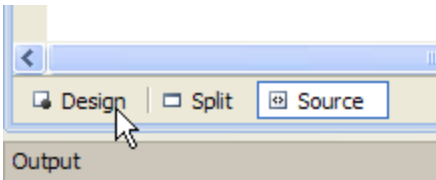
The Flash Web Viewer sample includes the following files to showcase the capabilities of the new FlashViewer ViewerType option in the Professional Edition WebViewer control. In the Solution Explorer, double-click a file to open it.

Run the project to explore the features.



FlashViewer.aspx

This file contains the WebViewer control. It opens to the Source view, so click **Design** at the bottom to see the WebViewer.



Click the WebViewer control at the top of the page to select it.



In the Properties window, notice that the **ViewerType** property is set to **FlashViewer**, and the **Height** and **Width** properties are set to **100%**. (This ensures that the viewer resizes to fill the browser window.) The **ReportName** property is not set, as the ReportName value is passed to it in code.



Expand the **FlashViewerOptions** node to see properties specifically related to this ViewerType. The **ResourceLocale** and **ThemeUrl** values are passed to the viewer in code, as are the **ShowSplitter**, **ShowThumbnails**, and **ShowToc** values.

Right-click on the design surface of FlashViewer.aspx and select **View Code**. In the C# or VB code page that appears, you can see the code used in the **Page_Load** event to pass the user input values into the WebViewer properties.

Two of the user input values are not included in the Properties window: **ShowToolBar** and **InsertButton**. The code demonstrates how to access the **FlashViewerToolBar.Visible** property, and how to create a custom button using the **FlashViewerToolBar.Tools.Add** method.

FlashViewerIntro.aspx

This file contains controls to collect user input, and a button to send the collected values to the WebViewer and open it. Right-click the file and select **View Code** to see the code used to populate the Themes drop-down list and to redirect to the FlashViewer form.

Reports folder

The Reports folder contains the three reports that you can select in the **Select Report** drop-down list. For detailed information on the Invoice report, see the **NEW Cross Section Control Sample**.

ActiveReports.FlashViewer.Resources.swf

The Resources file contains the six included locales that you can select in the **Select Language** drop-down list. For detailed information on Localization, see the **Localize the Flash Viewer** topic.

ActiveReports.FlashViewer.swf

The FlashViewer file contains the FlashViewer object, and must be copied into every FlashViewer project. Your original copy is stored in C:\Program Files\GrapeCity\ActiveReports 6\Deployment\Flash.

NEW Cross Section Control Sample

Tip: To easily select a control within the report, in the **Report Explorer**, expand the section node and select the control. The control is highlighted in the Report Explorer and on the report design surface.

This sample includes a **ViewerForm** with three tabs and three **Viewer** controls to highlight several new report features, and an **Invoice** report. Run the project to display the report in the viewer, and click the tabs to see the new features.

ViewerForm

The ViewerForm has three tabs, each with an ActiveReports Viewer control on it. Right-click the form and select **View Code** to see the code used to change the Invoice report's section properties at run time.

Select one of the Viewer controls and in the Properties window, expand the **Toolbar** property to see where the **Visible** property is set to **False**.

Invoice

The Invoice report demonstrates the usage of the following features:

PageHeader Section

- The **Shape** control provides a border around the Order ID and Order Date fields and labels.
- The OrderDate **TextBox** control has the **OutputFormat** property set to **d** to display a short date.
- The **Label** controls use the **BackColor**, **ForeColor**, and **Font** properties to add a distinctive style to the report.

GroupHeader Section

- The new **CrossSectionBox** control is hosted in the GroupHeader section, and spans the Detail section to end in the GroupFooter section, forming a rectangle around the details of the invoice at run time.

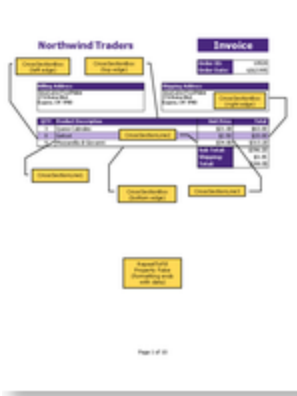


- Three of the new **CrossSectionLine** controls are hosted in the GroupHeader section, and span the Detail section to end in the GroupFooter section, forming vertical lines between columns of invoice details at run time.



Note: If you try to drop a cross-section control into a section other than a header or footer, the mouse pointer changes to Unavailable, and you cannot drop the control.





- Two of the TextBox controls use a **CalculatedField** in the **DataField** property.



Tip: In the Report Explorer, expand the **Fields** node, then **Calculated** to see all of the calculated fields.

Select **BillingAddress** or **ShippingAddress** to take a closer look at the **Formula** used in the Properties window.

- The **Line** control is used below the column header labels to draw a horizontal line across the width of the report. (It is not visible at design time unless you make the Height of the GroupHeader section larger.)
- The **DataField** property of the section is set to the **OrderID** field, so that the section (followed by related details and GroupFooter) prints once per order.

Detail Section

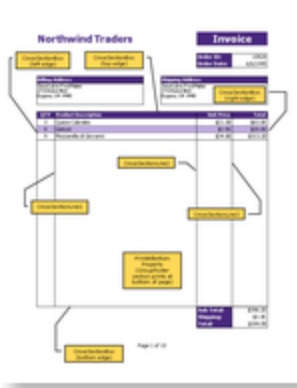
- Click the **Detail RepeatToFill** tab to see the Detail section with the new **RepeatToFill** property set to **True**. This ensures that the formatting (alternating purple and white rows and CrossSection controls) fills space as needed to push the GroupFooter section to the bottom of the page, just above the PageFooter section.



- Four **TextBox** controls display each row of data associated with the current GroupHeader OrderID.
- The **OutputFormat** property of the UnitPrice and Total fields is set to **C** to display currency.
- The **Line** control is used below the text boxes to draw horizontal lines across the width of the report under each row of data. (It is not visible at design time unless you make the Height of the Detail section larger.)
- Right-click the report and select **View Code** to see the code used in the **Detail Format** event to create a **green bar** (or in this case, purple bar) report by alternating the **BackColor** property of the section.
- Click the **Data Source** icon on the **Detail** band to review the **Connection String** and **SQL Query** used in the report.

GroupFooter Section

- Select the **GroupFooter PrintAtBottom** tab to see the GroupFooter section with the **new PrintAtBottom property** set to **True**. This pulls the GroupFooter section to the bottom of the page, just above the PageFooter section. Run the project and click the PrintAtBottom tab to see this feature in action.

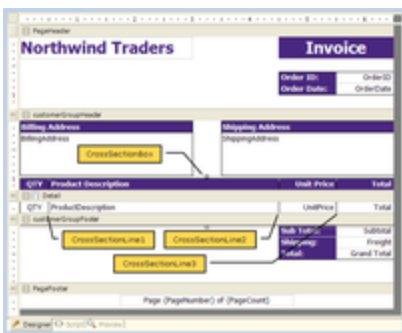


- This section also has the **NewPage** property set to **After** so that a new page is printed for each OrderID (the associated GroupHeader's DataField).
- The Subtotal text box uses the following properties:
 - The **DataField** property uses a **CalculatedField**.
 - The **SummaryFunc** property is set to **Sum**, to add the values of the field in the detail section.
 - The **SummaryGroup** property is set to the name of the **GroupHeader**, to reset the summary value each time the GroupHeader section runs.
 - The **SummaryRunning** property is set to **Group** so that the value accumulates for the group rather than for the entire report or not at all.
 - The **SummaryType** property is set to **GrandTotal**.
- Right-click the report and select **View Code** to see the code used in the **GroupFooter Format** event to calculate the value for the Grand Total text box, and to format it as currency.

PageFooter Section

- The **ReportInfo** control uses a **FormatString** property value of **Page {PageNumber} of {PageCount}**, one of the preset values you can use for quick page numbering.

Design Time



NEW Style Sheets Sample

This sample demonstrates how you can change styles at run time for a different look with the same report. The project includes two reports, three reportstyles, and a form with the ActiveReports Viewer control and several other controls that allow you to select any combination of styles and reports.

Report Style Sheets

Look in Solution Explorer to see several *.reportstyle files. These are XML-based files that hold styles that you can apply to TextBox, Label, CheckBox, and ReportInfo controls on ActiveReports. Double-click one to open it. Each reportstyle contains a set of values for each of the standard style names:

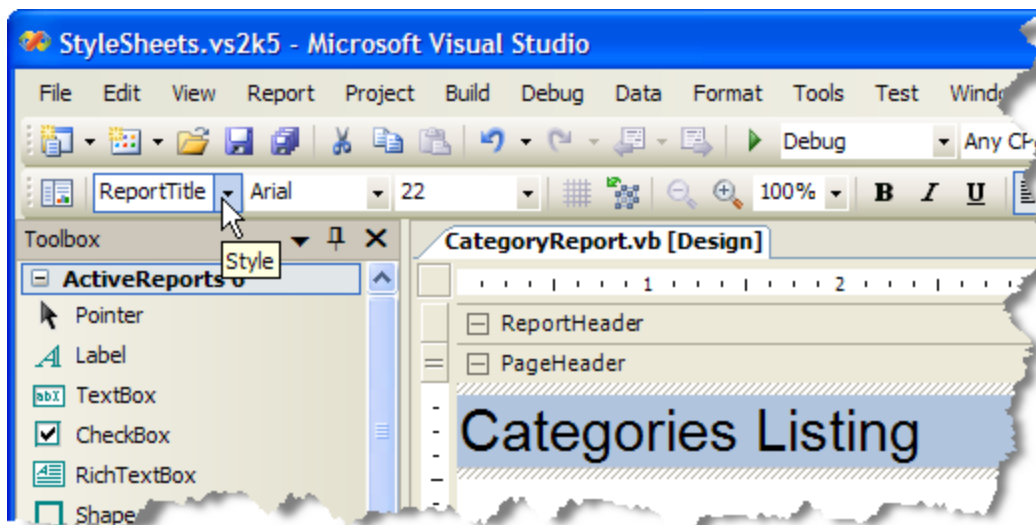
- Normal
- Heading1
- Heading2
- Heading3
- DetailRecord
- ReportTitle

When you select one of these style names on a report control, ActiveReports retrieves the style values, such as font size and color, from the specified style sheet when it runs the report.

For more information on creating your own style sheets, see [Use External Style Sheets](#).

Reports

Two reports, **CategoryReport** and **ProductsReport**, are included in this sample so that you can each of the styles applied in different ways. Open one of the reports, and select the TextBox and Label controls on it to see which style is used for each.



StyleSheetsForm

The form in this project features radio buttons for choosing the report and style you want, a **Choose** button that opens a standard Windows Open dialog where you can select a reportstyle, and a **Run report** button that runs the selected report, applies the selected reportstyle, and displays the results in the ActiveReports viewer control below.

To see how all of this works, right-click the form and select **View Code**.

Choose Button Click Event

This event contains code that sets up an Open dialog that shows only *.reportstyle files, and passes the selected reportstyle path and file name string to the externalStyleSheet variable.

Run Report Button Click Event

This event contains code that creates an empty ActiveReport object, assigns the selected report to it, and assigns a path and file name string to the styleSheet variable. It then assigns the style sheet to the report using the `LoadStyles(styleSheet)` method, runs the report, and displays it in the viewer.

Annual Report Sample

The Annual Report sample demonstrates the use of subreports, section properties, and the Chart control. See a description of each file below.

AnnualReport

This is the main ActiveReport for the project.

ReportHeader Section

This report features a two-page ReportHeader section that uses a **PageBreak** control to separate the two pages, and breaks to the second page by setting the ReportHeader section's **NewPage** property to **After**. This report shows how you can use the **BackColor** and **ForeColor** properties of labels to create a distinctive look for your reports.

The ReportHeader section also has a **SubReport** control that links to the ProductSalesByCategory report in the code behind the report, in the **ReportStart** event.



Best practice: It is a best practice to initialize reports in the **ReportStart** event rather than a section **Format** event so that a new report is not initialized every time the section runs.

If the SubReport control were in a section that prints multiple times, you would need to assign the report in the section Format event while still initializing in the ReportStart event. See the SubReports sample for more information.

The yellow background in the right half of the ReportHeader below the page break is achieved by using the **Shape** control and setting its **BackColor** property. The image to the left is a **Picture** control.

Detail Section

The Detail section contains two **SubReport** controls that link in the code behind the report to the Top10Customers and Top10Products reports. In most reports, the Detail section would run multiple times. In this report, the Detail section has only labels, and no bound textboxes, so it will only run once. Therefore, the Top10 reports can be assigned to the SubReport control in the **ReportStart** event where it is initialized.


Notice that the **ReportFooter** section has its **Height** property set to **0**. This is because, except for the Detail section, all sections come in pairs. In order to use the ReportHeader section, you must also have a ReportFooter section. If you do not want to use it, you can set its Height to 0.

GetDBPath

This is a helper class that finds the installation path of ActiveReports, and extrapolates the path of the sample NorthWind database that is included in the installation. It returns a connection string that is used by all of the ActiveReports samples that use the NorthWind database.

ProductSalesByCategory

This is the ActiveReport that is assigned to the **SubReport** control in the ReportHeader section of the Annual Report.

 **Best practice:** Notice that this report has had its ReportHeader/Footer and PageHeader/Footer sections removed. That is a best practice for reports to be used as subreports. These sections are not printed within the SubReport control, so removing the sections saves on processing.

Notice also that the **PrintWidth** property of this report is only **2.677** inches. This is so that it fits easily within the SubReport control on the Annual Report.

This report uses the **GroupHeader** section to display labels for the data fields that fill the **Detail** section. The fields in the Detail section repeat once for each row of data in the database.

Right-click in the grey area around the report and select **View Code** to see the code that sets the data source for the report, and sets the background color to yellow on every second row of data.

StartupForm

This form contains the ActiveReports **Viewer** control. The **Dock** property of the viewer is set to **Fill** so that it resizes automatically with the form at run time. Right-click and select **View Code** to see the code that displays the AnnualReport when the form loads.

Top10Customers and Top10Products

The Top10Customers and Top10Products reports use only two sections, GroupHeader and Detail. The **PrintWidth** property of each report is set to **3.135** inches so that it fits into the subreport control on the Annual Report.

The GroupHeader section of each report is filled with a **Chart** control. Click the chart to see its properties in the Properties window. At the bottom of the Properties window, click the **Data Source** verb to open the Chart Data Source dialog. In the **Query** box, you can see the SQL query that selects the top 10. For more information on creating charts, see **Chart Walkthroughs**.

The Detail section of each report has two bound **TextBoxes** and a **Label** control. Right-click and select **View Code** to see the code that sets the data source for the report, passes data to the Chart control, alternates the background color of the detail section, and sets the **Text** property of the label.

Bound Data Sample

The Bound Data sample demonstrates the use of seven different data binding techniques. The sample Invoice report demonstrates the use of grouping and of summary functions.

MainForm

The MainForm uses the ActiveReports **Viewer** control in the bottom section of the form, and a panel docked to the top contains seven tabs, each with a different data binding technique. Click to select a tab, and then double-click the button on the tab to jump to the button's **Click** event in the code.


- Bind to DataSet
- Bind to DataReader
- Bind to DataView
- Bind to DataTable
- Bind to SQL Server
- Bind to OleDb
- Bind to XML

Above the **Data Binding Code** region is the **Drop Down Population Code** region that is used to populate the combo boxes on the DataView and SQL Server tabs. The XML tab also features a button that generates a DataSet and saves it as an XML data file.

Run the project and click to select a tab, then click the buttons to check the functionality of each type of data binding as it processes the data, passes it to the report, and displays it in the viewer below.

Invoice

The Invoice report uses three GroupHeader sections, the Detail section and a GroupFooter section to display data, and adds a label in the PageFooter section.

 **Note:** Except for the Detail section, all sections come in header and footer pairs. Unused counterparts to the sections in use have their **Height** properties set to **0** and their **Visible** properties set to **False**.

ghOrderHeader

The **DataField** property of this section is set to **OrderID**. This setting, in conjunction with data ordered by the OrderID field, causes the report to print all of the information for one order ID value, including all of the related details and footers, before moving on to the next order ID. For more information on grouping, see **Grouping Data**.

This section contains a Picture control, a number of Label controls, and two bound TextBox controls. The TextBoxes are bound using the **DataField** property in the Properties window, and the date is formatted using the **OutputFormat** property.

ghOrderID

The **DataField** property of this section is also set to **OrderID**. This allows subtotal summary functions in the related GFOOrderID section to calculate properly.

This section contains a number of labels and bound text boxes, as well as two **Line** controls.

ghTableHeader

This section contains only labels for the data to follow in the Detail section.

Detail

This section contains bound TextBox controls. These render once for each row of data found in the current OrderID before the report moves on to the GroupFooter sections.

GFOOrderID

The **NewPage** property of this section is set to **After**. This causes the report to break to a new page and generate a new invoice after this section prints its subtotals.

This section contains several labels and several text boxes. Two of the TextBox controls use the following properties to summarize the detail data: **SummaryFunc**, **SummaryGroup**, and **SummaryType**. For more information, see **Create Summary Fields**.

The **Total** TextBox does not use the DataField property or any of the summary properties, or even any code behind the report. To find the functionality of this text box, click the **Script** tab at the bottom of the report.



For more information on using Script with ActiveReports, see **Scripting**.

PageFooter

This section has one simple Label control. Since none of the GroupFooter sections has its **PrintAtBottom** property set to **True**, the PageFooter prints at the bottom of each page of the report. For more information about report sections and the order in which they print, see **Report Structure** and **Section Events**.

Category Selection Sample

This sample shows you how to pass a SQL string into a report at run time. It consists of a CategoryProducts report and a CategorySelectForm.

CategorySelectForm

The ActiveReports Viewer control fills most of the form, and a combo box at the top allows the user to select which data to send to the viewer.

Right-click the form and select **View Code** to see how this is done.

- The **getCategories** code populates the combo box.
- The **runCategoryReport** code runs the report with a SQL string with the CategoryName passed in by the combo box selection.
- The **SelectIndexChanged** event calls runCategoryReport and passes it the CategoryName.

CategoryProducts Report

This report lists products in the selected category, and summarizes the number of products. Here are the features used in each section:

ReportHeader

This section cannot be deleted, because the related ReportFooter section is used. When this is the case, the best practice is to set the unused section's **Height** property to **0**.

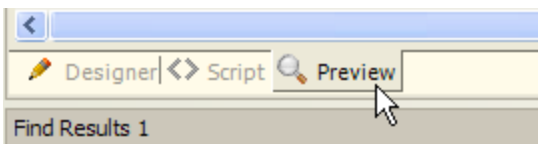
PageHeader

- This section uses a **Label** control with a large **Font** size and its **BackColor** property set to **LightSteelBlue** for the report title.
- The **TextBox** control, **txtCategory**, has its **DataField** property set to **CategoryName**.
- Two **Line** controls and two more **Label** controls create the column headers for the report.

Detail

This section contains two bound **TextBox** controls to display each product in the selected category along with its price.

Although the form passes data to the report at run time, the report's data source is set for design time use. It is easier at design time to drag bound fields onto the report from the **Report Explorer** than it is to create them and set their properties. The data source also allows you to preview the report while you are designing it.



Click the **DataSource Icon** on the Detail band to view the data source.



PageFooter

This section uses the **ReportInfo** control to display Page N of M. In the Properties window, you can select a way to display the page number and run date in the **FormatString** property.

ReportFooter

This section contains a **Label** control and a bound **TextBox**. The TextBox uses the **SummaryType** of **GrandTotal** to display the total number of products in the selected category.

Charting Sample

The Charting sample consists of a form with the ActiveReports Viewer control to display the report, and an ActiveReport with two Chart controls, one bound and one unbound.

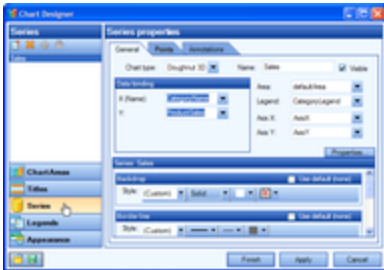
rptCharting

The **GroupHeader** section has its **NewPage** property set to **After**. This causes the Detail section to print on a new page.

ChartSalesCategories

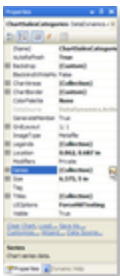
The chart in the GroupHeader section gets its data from the chart data source. To see the data source, select the chart, and in the verbs section of the Properties window, click **Data Source**. If you have trouble with this, see **Access the Chart Wizard and Data Source** for help.

Also in the verbs section of the Properties window, click **Customize** to open the **Chart Designer** window. Click the **Series** tab to see the fields bound to the X and Y axes. Also on that tab, you can see that the **Chart type** is set to **Doughnut 3D**. Scroll down to see the **Marker** settings that label each slice of the pie in the chart.



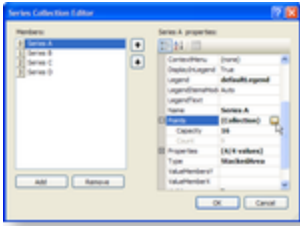
Click the **Legends** tab to see which settings are used to set up the legend at the bottom of the chart. The **Titles** tab is where the title at the top of the report is set, and the **Appearance** tab is where the colors of the chart are set.

You can also access all of these properties in the Properties window by clicking the ellipsis button that appears when you select the property.

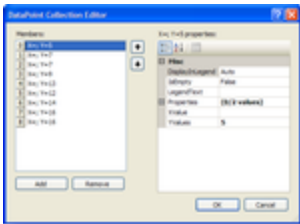


ChartUnbound

The chart in the Detail section is not connected to data. Instead, values for data points are set in the Series Collection Editor dialog. To open the dialog, click the chart to select it and in the Properties window, click the Series property to display the ellipsis button (see image above). Click the ellipsis button to open the dialog.



Here you can see the four series in the chart, A through D. Scroll down in the properties for one of the series, and you can see that the chart **Type** (in this case, StackedArea) can be set for each series. Select the **Points** property. Click the ellipsis button that appears next to it to open the DataPoint Collection Editor dialog where the Y value for each data point is entered.



Close both dialogs to return to the chart, and in the Properties window, expand the **Backdrop** property to see the settings used to create the gradient blue backdrop for the chart. Click the ellipsis button next to the **Titles** property to find where the header and footer titles are set.

For more information on creating charts, see the **Create Charts** topic and the **Chart Walkthroughs** section.

ViewerForm

The ViewerForm contains the ActiveReports **Viewer** control, with its **Dock** property set to **Fill**. This enables the viewer to automatically resize along with the form. Right-click the form and select **View Code** to see the three lines of code used to run the report and display it in the viewer.

Cross Tab Report Sample

The Cross Tab Report Sample consists of a StartForm with an ActiveReports Viewer control and a ProductWeeklySales report.

StartForm

The Viewer control has its **Dock** property set to **Fill**. This ensures that the viewer resizes along with the form at run time. Right-click the form and select **View Code** to see the code used to run the report and display it in the viewer.

ProductWeeklySales

This report features a number of accumulated values using summary function property settings and calculated in the code behind the report.

ReportHeader

This section of the report features static controls including Labels, a Picture, a Line, and a **Shape** control with its **BackColor** property set to yellow. The report header prints only once, on the first page of the report, so it is a good place for a title, company information, and a logo.

PageHeader

The page header section also contains static Label controls that print at the top of each page and serve as column headers for the group header sections.

ghCategory

This group header section has its **DataField** property set to **CategoryName**. This setting, along with data sorted by the same field, produces a report grouped by category. The section contains one bound TextBox control to display the category name at the beginning of each group. The section's **UnderlayNext** property is set to **True** so that the category prints to the left of the top line of data instead of above it.

ghProduct

Although this group header contains no controls and is hidden using the **Height** and **Visible** properties, it still performs two important functions. First, its **DataField** property is set to **ProductName**, sorting the data inside each category by product, and second, its related group footer section displays the bulk of the data for the report.

Detail

The detail section of this report is hidden using the **Height** and **Visible** properties, but it does contain four bound fields whose values are used in the code behind the report.

Code

ReportStart Event

Right-click the report and select **View Code** to see how this report, which does not have its data source set using the data source icon on the Detail band, gets its data. Private variables are created to hold values, and are set initially within the **ReportStart** event.

DataInitialize Event

The data source is set in the **DataInitialize** event, and then unbound fields are added to the report's **Fields** collection. For more information on unbound reporting, see **Unbound Reporting**.

FetchData Event

The **FetchData** event, which runs once for each row in your dataset, is where most of this report's logic is set. See the comments in the code to understand how data is calculated and passed to the report's **Fields** collection.

Detail Format Event

In the **Detail Format** event, the value from the hidden **txtDetProduct** text box is collected and passed to the **_sProductName** variable.

For more information on section events, see **Section Events**.

gfCategory Format Event

In the **gfCategory Format** event for the outer group footer section, the **Value** for the **txtCatPQTDChange** text box is calculated by subtracting the prior year's quarter-to-date sales figure from the current quarter-to-date sales figure.

The **BackColor** of the **txtCatPQTDChange** text box is set to **Red** if the value is negative.

gfProduct Format Event

In the **gfProduct Format** event for the inner group footer section, the product name collected from the Detail Format event is passed to the **txtProduct** text box. The **Value** for the **txtPQTDChange** text box is calculated by subtracting the prior year's quarter-to-date sales figure from the current quarter-to-date sales figure.

The **BackColor** of the **txtPQTDChange** text box is set to **Red** if the value is negative.

gfProduct

This group footer section displays the bulk of the data for the report in TextBox controls that have values passed in code, or are bound to fields from the report's **Fields** collection (see `FetchData` and `DataInitialize` events in the code) using the **DataField** property. The total units and sales for each product is summarized using the following properties:

- **SummaryFunc**: Sum (the default value) adds values rather than counting or averaging them.
- **SummaryGroup**: `ghProduct` summarizes the values that fall within the current product group.
- **SummaryRunning**: None (the default value) ensures that this value is reset each time the product group changes.
- **SummaryType**: `SubTotal` summarizes the current group rather than a page or report total.

gfCategory

This group footer section displays totals of the `gfProduct` data in TextBox controls that have values passed in code, or are bound to fields from the report's **Fields** collection (see `FetchData` and `DataInitialize` events in the code) using the **DataField** property. The total units and sales for each category is summarized using the following properties:

- **SummaryFunc**: Sum (the default value) adds values rather than counting or averaging them.
- **SummaryGroup**: `ghCategory` summarizes the values that fall within the current category group.
- **SummaryRunning**: None (the default value) ensures that this value is reset each time the category group changes.
- **SummaryType**: `SubTotal` summarizes the current group rather than a page or report total.

PageFooter

This section is not used, so it is hidden using the **Height** and **Visible** properties. Otherwise, it would print at the bottom of each page. The section cannot be deleted, because its related `PageHeader` section is in use.

ReportFooter

This section is not used, so it is hidden using the **Height** and **Visible** properties. Otherwise, it would print once at the end of the report. The section cannot be deleted, because its related `ReportHeader` section is in use.

Custom Preview Sample

The Custom Preview sample consists of a parent `CustomPreviewForm` with menus, a child `PreviewForm` with an ActiveReports Viewer control, an `ExportForm` with a `PropertyGrid`, a `Reports` folder with six reports, and a `Resources` folder with six icons.

CustomPreviewForm

The `CustomPreviewForm` has its **IsMdiContainer** property set to **True**. This ensures that when a user opens a child `PreviewForm`, it is contained within the parent `CustomPreviewForm`.

This form has a menu bar, **mnuMain**, with three menus: **File**, **Reports**, and **Window**. The **MergeType** property of the File menu is set to **MergeItems** so that the menu items from any child `PreviewForms` are added to it. The form also has two dialogs: **dlgOpenFile**, and **dlgPrint**.

Right-click the form and select **View Code** to see how reports selected from the `Reports` menu are run and passed to the `PreviewForm` using the `ShowReport` code.

PreviewForm

The `PreviewForm` has the ActiveReports Viewer control with its **Dock** property set to **Fill** so that the viewer resizes with the form at run time. The form also has a File menu with its **MergeType** set to **MergeItems** so that when a report is open, the File menu on the `CustomPreviewForm` displays the **Export**, **Save**, and **PrinterSetup** menu items. The form also has one dialog: **dlgPrint**.

Right-click the form and select **View Code** to see how, in the **PreviewForm Load** event, two custom buttons are added to the toolbar. The **ToolClick** event of the viewer calls the **SaveDocument** and **ExportDocument** functions for the new buttons. The menu item click events call the same functions for the related menu items.

The **SaveDocument** function opens the dialog **dlgSave**, while the **ExportDocument** opens a new **ExportForm**.

ExportForm

The Export Report Document form opens from the **ExportDocument** function on the **PreviewForm**. The form features an Export Format combo box, **cboExportFormat**, that populates the **PropertyGrid** control below based on the selected item. The export types are added to **cboExportFormat** via the **Items (collection)** property.

Right-click the form and select **View Code** to see how, in the **cboExportFormat SelectedIndexChanged** event, the property grid control's **SelectedObject** is set to the selected export. This ensures that only the properties related to each export type show in the grid.

See the **btnOK Click** event for the code that exports the report to the selected file name and format, and the **btnSaveFile Click** event for the code that opens the Save dialog.

Resources Folder

This folder holds the icons used in adding tools to the toolbar.

Reports Folder

Most of the reports in the Reports folder are documented in more detail elsewhere. Here, they are used mainly to show the Viewer and Export features.

- **Catalog** is documented below.
- **CustomerLabels** is explained in the **Address Labels** walkthrough.
- **EmployeeProfiles** is explained in detail in the code behind the report. (Right-click and select **View Code**.)
- **EmployeeSales** is explained in the **Bar Chart** walkthrough.
- **Invoice** is in the **Bound Data Sample**.
- **Letter** is explained in the **Mail Merge with RichText** walkthrough.

Catalog Report

The Catalog report uses the **PageBreak** control and the **NewPage** property to create a cover at the beginning and an order form at the end of the catalog. It uses grouping to list products by category.

ReportHeader

In the top of the **ReportHeader** section, the **Picture**, **Line**, and **Label** controls are used to create a static cover page for the catalog. The **PageBreak** control allows a second page of static labels to be set up in the same section, and setting the section's **NewPage** property to **After** ensures that the report breaks to a new page before rendering the next section. The **ReportHeader** section prints once per report.

PageHeader

This section is not in use, so the **Height** property is set to **0**. This section cannot be deleted because its related **PageFooter** section is in use.

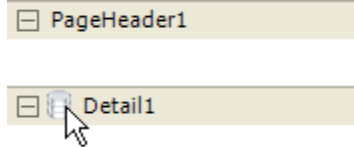
ghCategoryName

This **GroupHeader** section has its **DataField** property set to **CategoryName**. This, along with sorting the data by the same field, ensures that all of the details for one category are printed before the report moves on to the next category. Also, the section's **GroupKeepTogether** property is set to **All**. This causes ActiveReports to attempt to print the group header, related details, and the group footer together all on one page.

The controls in this section include two bound TextBox controls and a bound Picture control, along with a row of labels to serve as column headers for the Detail section to follow.

Detail

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the Connection String and SQL Query that provide data for the bound fields.



The Detail section has four bound **TextBox** controls. Select one of them and you can see in the Properties window the field that it is bound to in the **DataField** property. The **UnitPrice** text box also uses the **OutputFormat** to display the data in currency format. This section prints once for each row of data.

gfCategoryName

This section is used only to render a horizontal Line control after each category grouping is completed.

PageFooter

This section is used to display the page number at the bottom of each page.

ReportFooter

This section has the **NewPage** property set to **Before** to ensure that it begins at the top of a new page.

Label, Shape, and Line controls are used to create the static order form layout in this page-long report section that prints once at the end of the report.

Hyperlinks and Drill Down Sample

This sample consists of three reports and a ViewerForm. The reports use the Hyperlink event of the viewer to pass a value from the Hyperlink property of a TextBox control to a Parameter value in a more detailed report.

ViewerForm

This form contains only an ActiveReports Viewer control with its **Dock** property set to **Fill**. This ensures that the viewer resizes along with the form at run time.

Right-click the form and select **View Code** to see the code that allows multiple ViewerForms to display, and see the **Form Load** event for the code that loads the main report into the viewer.

See the **Viewer Hyperlink** event for the code that collects a string value from the **Hyperlink** property of the clicked TextBox on the main report and passes it into the customerID Parameter of the report **DrillDown1**, or collects a numeric value and passes it to the orderID Parameter of the report **DrillDown2**. This code then runs the report with the parameter value and displays it in another instance of the ViewerForm.

DrillDownMain Report

The main report that is loaded in the ViewerForm by default uses the PageHeader and Detail sections.

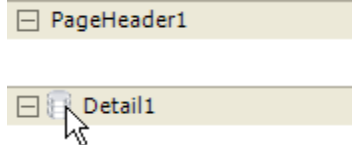
PageHeader Section

This section contains three Label controls to serve as column headers for the details, and a CrossSectionBox control. For more information, see **Cross Section Controls**.

Detail Section


The Detail section has its **BackColor** property set to **Thistle**, and its **RepeatToFill** property set to **True**. This ensures that the background color reaches all the way to the bottom of the page when there is not enough data to fill it.

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the Connection String and SQL Query that provide data for the bound fields.



The Detail section has three bound **TextBox** controls that display a list of customer information. Select **CustomerID** and you will see that the **HyperLink** property is not set in the Properties window. To see the code that assigns the data from the TextBox to its HyperLink property, right-click the report and select **View Code**.

The **HyperLink** property is set in the **Detail BeforePrint** event. For more information, see **Section Events**.

 **Note:** This hyperlink does not work in Preview mode, because it relies on code in the ViewerForm to pass the value to DrillDown1 report's parameter.

PageFooter Section

This section is not in use, so it is hidden by setting the **Visible** property to **False**. This section cannot be deleted, because its related PageHeader section is in use.

DrillDown1 Report

This report looks similar to the DrillDownMain report, but the main difference is that it has a CustomerID parameter in its SQL Query.

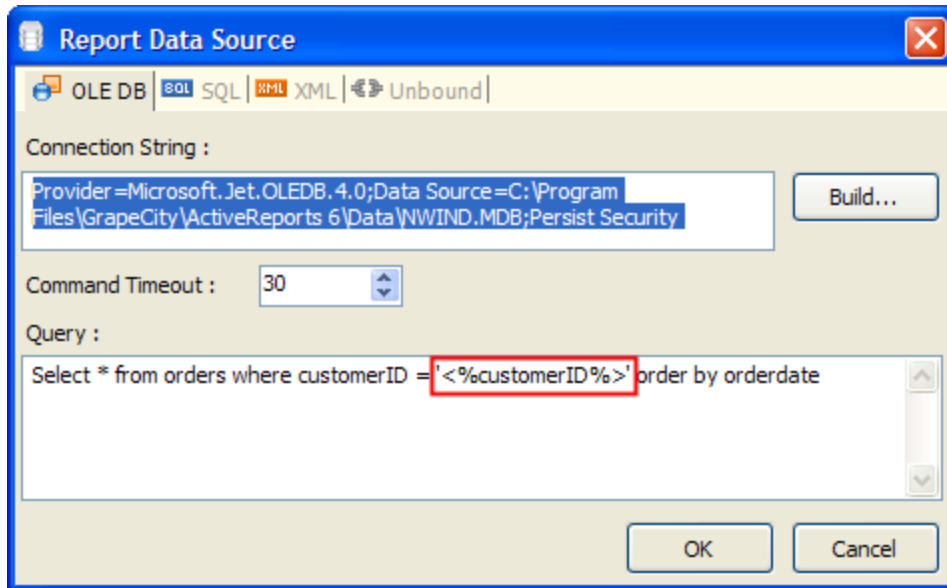
GroupHeader Section

Since this report only displays order information for the CustomerID from the clicked hyperlink, the PageHeader section could have been used, but this report uses the GroupHeader section. To make this section print at the top of each page, its **RepeatStyle** property is set to **OnPage**.

Like in the previous report, this section contains three Label controls to serve as column headers for the details, and a CrossSectionBox control.

Detail Section

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the parameter in the SQL Query that collects its value from the ViewerForm.



Parameters in SQL Queries are denoted by the <% and %> symbols that trigger ActiveReports to add them to the report's Parameters collection. For more information, see **Parameters**.

The Detail section has five bound TextBox controls that display a list of order information for the customer. Select **OrderID** and you will see that the **HyperLink** property is not set in the Properties window. To see the code that assigns the data from the TextBox to its HyperLink property, right-click the report and select **View Code**.

The **HyperLink** property is set in the **Detail BeforePrint** event. For more information, see **Section Events**.

GroupFooter Section

This section is not in use, so it is hidden by setting the **Visible** property to **False**. This section cannot be deleted, because its related GroupHeader section is in use.

DrillDown2 Report

Like DrillDown1, this report has a parameter in its SQL Query, but unlike the other two reports, this one has no hyperlink. It displays order details for the OrderID value passed into it from the clicked hyperlink in DrillDown1.

GroupHeader Section

Like in the previous report, this section contains Label controls to serve as column headers for the details, and a CrossSectionBox control.

Detail Section

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the parameter in the SQL Query that collects its value from the ViewerForm.

Rdf Viewer Sample

The RDF Viewer sample consists of an RdfViewerForm with an ActiveReports Viewer control, and an ExportForm with a Property Grid. The sample also contains an RDFs folder of saved reports.

RdfViewerForm

This form contains an ActiveReports Viewer control with its **Dock** property set to **Fill**. This ensures that the viewer

resizes along with the form at run time.

It also contains a MenuStrip and an OpenFileDialog. To see the code for each of these, right-click the form and select **View Code**.

Code

The **GetReportDoc** property gets the Document object from the viewer. The ExportForm uses this property to export report documents.

The **OpenToolStripMenuItem Click** event filters to show only RDF files and opens the Open File dialog to the RDFs folder. The **dlgOpenFile FileOK** function loads the selected RDF file into the viewer.

The **ExportToolStripMenuItem Click** event opens a new ExportForm. Each of the other menu item click events performs its function in a straightforward manner.

For more information on the Viewer control, see **Viewing Reports**.

ExportForm

This form contains a combo box to collect the user-selected export format, a property grid to display properties for the selected format, and an OK button to export the report to the selected format. It also contains a Save dialog. Right-click the form and select **View Code** to see how this is done.

Code

The overloaded **Show** method allows the ExportForm to be called as a child object of the RdfViewerForm.

The Export Format combo box **SelectIndexChanged** event sets the **exportComponent** variable and the pgOptions property grid's **SelectedObject** to the selected export type.

The **exportComponent** variable is picked up in the OK button **Click** event, and then the report Document is pulled from the viewer and exported to the selected format.

RDFs Folder

An RDF file is a static copy of a report saved to the native Report Document Format. This can be loaded into the viewer without running it or accessing data. For more information, see **Save and Load Report Files (RDF)**.

The following five reports are included in this sample:

- Catalog.rdf
- Customer Labels.rdf
- Employee Profiles.rdf
- Employee Sales.rdf
- Invoice.rdf
- Letter.rdf

SubReports Sample

The SubReports sample consists of a StartForm with an ActiveReports Viewer control with two buttons, one to run and load each main report, and four reports.

StartForm

The StartForm has a Viewer control with its **Dock** property set to **Fill**. This ensures that the viewer resizes along with the form at run time. It also has a panel docked to the top that holds two buttons to run the two main reports. Right-click the form and select **View Code** to see the code that accomplishes this.

Code

The Click event of the **Run SubReport** button runs a new **Customers** report and loads it into the viewer. The Click event of the **Orders Report** button runs a new **Orders** report and loads it into the viewer.

Customers Report

The Customers report uses only the Detail section. It uses several Label controls and bound TextBox controls to display customer data, and a SubReport control to display details about the current customer's orders. This section renders once for each line of data found. The data source icon is not used in this report, as the data, along with the subreport, is set up in code. Right-click the report and select **View Code** to see the code that accomplishes this.

Code

In the **ReportStart** event, a new instance of the Orders report is created. This event is the most efficient place to instantiate reports for use in subreport controls as it fires only once when the report is run.

The **DataInitialize** event is where the data source and SQL query for the main report is set. The data source for the subreport is set in the **Detail Format** event.

For more information on subreport usage, see **Subreports**.

Orders Report

The Orders report is displayed in the SubReport control in the Customers report. The **PageHeader** and **PageFooter** sections have been removed because these sections do not display in the SubReport control. This avoids the use of processing time and resources for sections that do not render.

This report uses the GroupHeader section to display static labels for the data in the Detail section. The Detail section displays all of the order data for the current customer in the main report.

OrdersMasterReport

The OrdersMasterReport displays general information about orders, and uses a subreport to display the order details.

ReportHeader Section

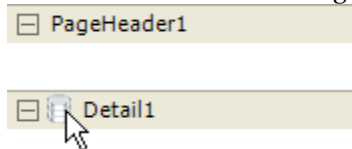
This section uses static **Label** controls to display labels for the details to follow.

PageHeader Section

Since this section is not in use, its **Height** property is set to **0**. The section cannot be deleted because the related PageFooter section is in use.

Detail Section

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the Connection String and SQL Query that provide data for the bound fields.



The Detail section has six bound **TextBox** controls that display a list of order information, and one **SubReport** control that displays the OrdersDetailReport.

PageFooter Section

This section, which renders once per page, uses the **ReportInfo** control to display page *n* of *m* at the bottom of each page.

ReportFooter Section

This section is not used, but cannot be deleted because the related ReportHeader section is in use.

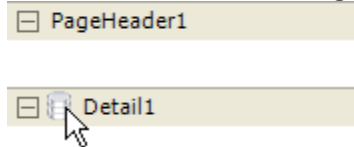
Code

The report for the SubReport control, OrdersDetailReport, is instantiated and assigned to the SubReport control in the **ReportStart** event.

OrdersDetailReport

This report has static Label controls in the ReportHeader section, and bound TextBoxes in the Detail section.

Click the gray report DataSource icon on the Detail section band to open the **Report Data Source** dialog, where you can see the Connection String and SQL Query that provide data for the bound fields.



Walkthroughs

The Walkthroughs section of the User Guide provides you with step-by-step tutorials that you can follow as you create projects in Visual Studio. The walkthroughs progress from basic through advanced for Standard and Professional Editions of ActiveReports.

Standard Edition Walkthroughs

Basic Data Bound Reports

Basic XML-Based Reports (RPX)

Address Labels

Columnar Reports

Overlying Reports (Letterhead)

Chart Walkthroughs

Basic Spreadsheet with SpreadBuilder

Group On Unbound Fields

Subreport Walkthroughs

Hyperlinks for Simulated Drill-Down Reporting

Mail Merge with RichText

Run Time or Ad Hoc Reporting

Web Walkthroughs (Standard Edition)

Layout Files with Embedded Script

Professional Edition Walkthroughs

Creating a Basic End User Report Designer (Pro Edition)

Web Viewer (Pro Edition)

Flash Viewer

Customizing the Flash Viewer UI

Silverlight Viewer (Pro Edition)

Basic Data Bound Reports

In ActiveReports, the simplest reporting style is a tabular listing of fields from a data source.

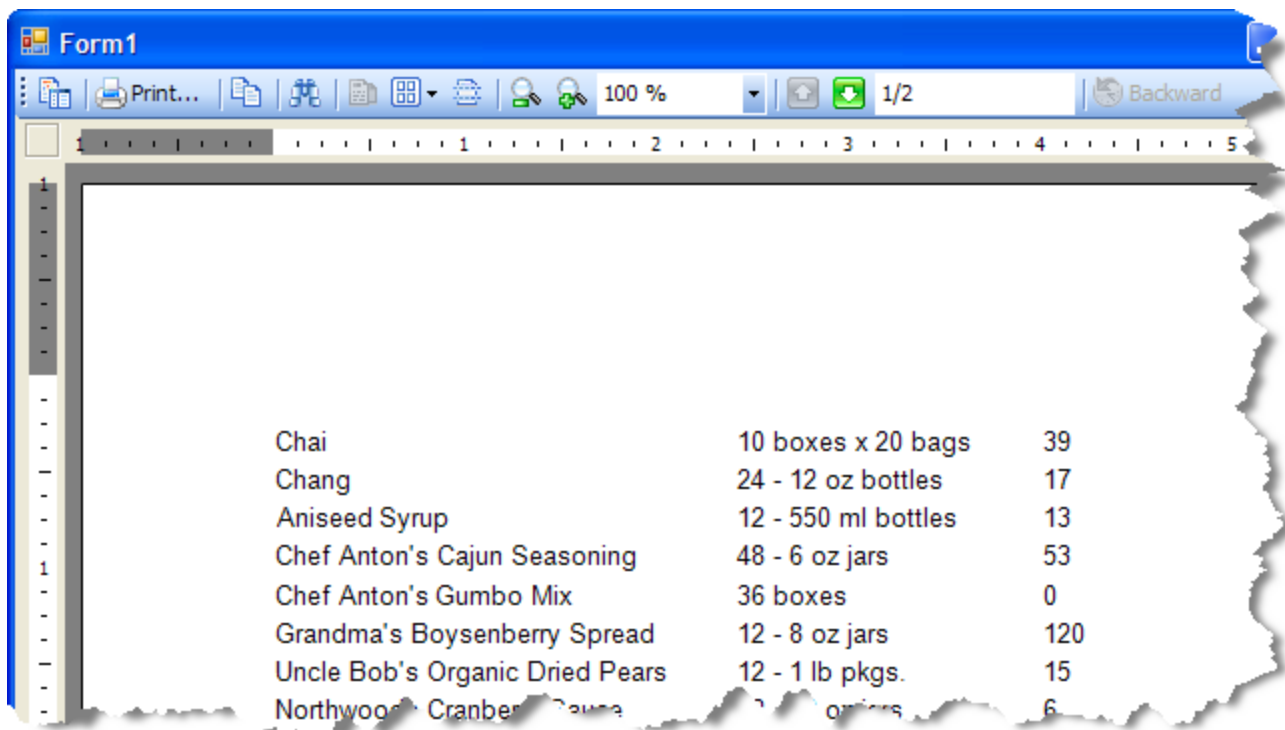
This walkthrough illustrates the basics of setting up bound reports by introducing the ideas of using the DataSource icon and dragging fields from the Report Explorer onto the report.

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 the report
- Viewing the report

To complete the walkthrough, you must have access to the Northwind database. A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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



| Product Name | Quantity | Price |
|---------------------------------|---------------------|-------|
| 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 |
| Northwood Cranberry Sauce | 2 - 6 oz jars | 6 |

To create a new Visual Studio project

1. Open Visual Studio.
2. From the **File** menu, select **New > Project**.
3. In the New Project dialog, select the project type in the Project Types section and then select **Windows Application** in the Templates section.
4. Change the name of your project and click **OK**.

To add an ActiveReport to the Visual Studio project

1. From the **Project** menu, select **Add New Item**.
2. Select **ActiveReports 6 (code-based) File** and rename the file rptBound.
3. Click **Add**.

To connect the report to a database

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



2. On the **OLE DB** tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Products
```

7. Click **OK** to save the data source and return to the report design surface.

To add controls to the report

1. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the detail section and set the properties of each textbox as indicated.

| Field | Text | Location | Size |
|-----------------|--------------|----------|----------|
| ProductName | Product Name | 0, 0 | 2,3, 0.2 |
| QuantityPerUnit | Quantity | 2,4, 0 | 1,5, 0.2 |
| UnitsInStock | Stock | 4, 0 | 1, 0.2 |

2. Click just below the fields to select the Detail section, and in the Properties Window, set the **CanShrink** property to **True** to eliminate white space in the rendered report.

To view the report

You can quickly view your report at design time by clicking the **Preview** tab at the bottom of the designer.


1. Drag the ActiveReports viewer control from the Visual Studio toolbox onto the Windows Form and set its **Dock** property to **Fill**.
2. Double click the title bar of the form to create a Form_Load event, and add the code needed to run the report and display it in the viewer.

Run the report and display it in the viewer using Visual Basic.NET

```
Dim rpt As New rptBound
rpt.Run ()
Me.Viewer1.Document = rpt.Document
```

Run the report and display it in the viewer using C#

```
rptBound rpt = new rptBound();
rpt.Run ();
this.viewer1.Document = rpt.Document;
```

 **Note:** On a 64-bit Windows operating system, you need to change the build target platform prior to viewing reports in runtime. To do this, in the Visual Studio Solution Explorer right-click the project name and select **Properties**. In the window that opens, go to the **Compile** tab for Visual Basic, or the **Build** tab for C#, and set **Platform Target** to **x86**.


3. Run the project to display the report in the viewer.

Basic XML-Based Reports (RPX)

ActiveReports 6 allows you to create reports with embedded script and save them to the XML-based RPX file format. By embedding script in reports saved as RPX files, you can later load, run, and display reports directly in the viewer control without rebuilding the application. This walkthrough illustrates how to create a simple report, using the XML-based report template.


This walkthrough is split into the following activities:

- Adding controls to a report to display data
- Adding scripting to supply data for the controls
- Applying a green-bar effect to the Detail section of a report
- Loading an xml-based report from resources

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database. A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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



| Product Name | Quantity | Unit | Price |
|---------------------------------|---------------------|------|-------|
| 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 |

To add controls to the report

1. Add an **ActiveReports 6 (xml-based) File** to a Visual Basic project and name it **rptScript**.
2. In the Solution Explorer click the **rptScript.rpx** item and set the **Build Action** property to **Embedded Resource**.
3. Add the following controls to the Detail section:

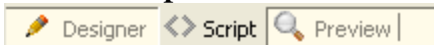
Detail section fields

| Control | Name | DataField | Location | Size |
|---------|--------------------|-----------------|-----------|-------------|
| TextBox | txtProductName | ProductName | 0, 0 in | 2.3, 0.2 in |
| TextBox | txtQuantityPerUnit | QuantityPerUnit | 2.4, 0 in | 1.5, 0.2 in |
| TextBox | txtUnitsInStock | UnitsInStock | 4, 0 in | 1, 0.2 in |

4. Click just below the fields to select the Detail section.
5. In the Properties Window, set the **CanShrink** property to **True** to eliminate white space in the rendered report.

To add scripting to the report to supply data for the controls

1. Click the **Script** tab located at the bottom edge of the report designer to access the scripting editor.



2. Add the scripting code.

The following example shows what the scripting code looks like.

Warning: Do not access the Fields collection outside the **DataInitialize** and **FetchData** events. Accessing the Fields collection outside of these events is not supported, and has unpredictable results.

To write the script in Visual Basic.NET

Visual Basic.NET script. Paste in the script editor window.

```

Private Shared m_cnn As System.Data.OleDb.OleDbConnection

Public Sub ActiveReport_ReportStart()
    'Set up a data connection for the report
    Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.mdb"
    Dim sqlString As String = "SELECT * FROM products"

    m_cnn = new System.Data.OleDb.OleDbConnection(connString)
    Dim m_Cmd As System.Data.OleDb.OleDbCommand = new
System.Data.OleDb.OleDbCommand(sqlString, m_cnn)

    If m_cnn.State = System.Data.ConnectionState.Closed Then
        m_cnn.Open
    End If
    rpt.DataSource = m_Cmd.ExecuteReader
End Sub

Public Sub ActiveReport_ReportEnd()
    'Close the data reader and connection
    m_cnn.Close
End Sub

```

To write the script in C#**C# script. Paste in the script editor window.**

```

private static System.Data.OleDb.OleDbConnection m_cnn;

public void ActiveReport_ReportStart()
{
    //Set up a data connection for the report
    string m_cnnString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.mdb";
    string sqlString = "SELECT * FROM products";
    m_cnn = new System.Data.OleDb.OleDbConnection(m_cnnString);
    System.Data.OleDb.OleDbCommand m_Cmd = new
System.Data.OleDb.OleDbCommand(sqlString, m_cnn);

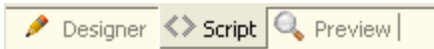
    if(m_cnn.State == System.Data.ConnectionState.Closed)
    {
        m_cnn.Open();
    }
    rpt.DataSource = m_Cmd.ExecuteReader();
}

public void ActiveReport_ReportEnd()
{
    //Close the data reader and connection
    m_cnn.Close();
}

```

To add scripting to alternate colors in the detail section

1. Click the **Script** tab located at the bottom edge of the report designer to access the scripting editor.



2. Add the scripting code.

The following example shows what the scripting code looks like.

To write the script in Visual Basic.NET

Visual Basic.NET script. Paste in the script editor window.

```
Dim b as boolean = true

Sub Detaill_Format
  if b then
    Me.Detaill.BackColor = Color.AliceBlue
    b= false
  else
    me.Detaill.BackColor = Color.Cyan
    b = true
  End If
End Sub
```

To write the script in C#

C# script. Paste in the script editor window.

```
bool color = true;
public void Detaill_Format()
{
  if(color)
  {
    this.Detaill.BackColor = System.Drawing.Color.AliceBlue;
    color = false;
  }
  else
  {
    this.Detaill.BackColor = System.Drawing.Color.Cyan;
    color = true;
  }
}
```

To view the report

You can quickly view your report at design time by clicking the **Preview** tab at the bottom of the designer.

1. Drag the ActiveReports viewer control from the Visual Studio toolbox onto the Windows Form and set its **Dock** property to **Fill**.
2. Double-click the title bar of the Windows Form containing the viewer to create a Form_Load event and add the code needed to load the RPX into a generic ActiveReport and display it in the viewer.

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

To write the script in Visual Basic.NET

Visual Basic.NET script. Paste INSIDE the form load event.

```
Dim asm As Reflection.Assembly = Reflection.Assembly.GetExecutingAssembly()
Dim st As IO.Stream = asm.GetManifestResourceStream(asm.GetName().Name +
".rptScript.rpx")
```



```
Dim report As New DataDynamics.ActiveReports.ActiveReport
Using reader As New System.Xml.XmlTextReader(st)
    report.LoadLayout(reader)
End Using
report.Run()
Viewer1.Document = report.Document
```

To write the script in C#

C# script. Paste INSIDE the form load event.

```
System.Reflection.Assembly asm = System.Reflection.Assembly.GetExecutingAssembly();
System.IO.Stream st = asm.GetManifestResourceStream(asm.GetName().Name +
".rptScript.rpx");
DataDynamics.ActiveReports.ActiveReport report = new
DataDynamics.ActiveReports.ActiveReport();
using (System.Xml.XmlTextReader reader = new System.Xml.XmlTextReader(st))
{
    report.LoadLayout(reader);
}
report.Run();
viewer1.Document = report.Document;
```

Address Labels

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

This walkthrough illustrates how to create a report that repeats labels using the `LayoutAction` property and prints labels to a laser printer. The labels in this example are 1" x 2.5" and print 30 labels per 8½" x 11" sheet.

The walkthrough is split up into the following activities:

- Connecting the report to a data source
- Adding controls to the report to display data
- Adding code to the `detail_Format` event to repeat labels

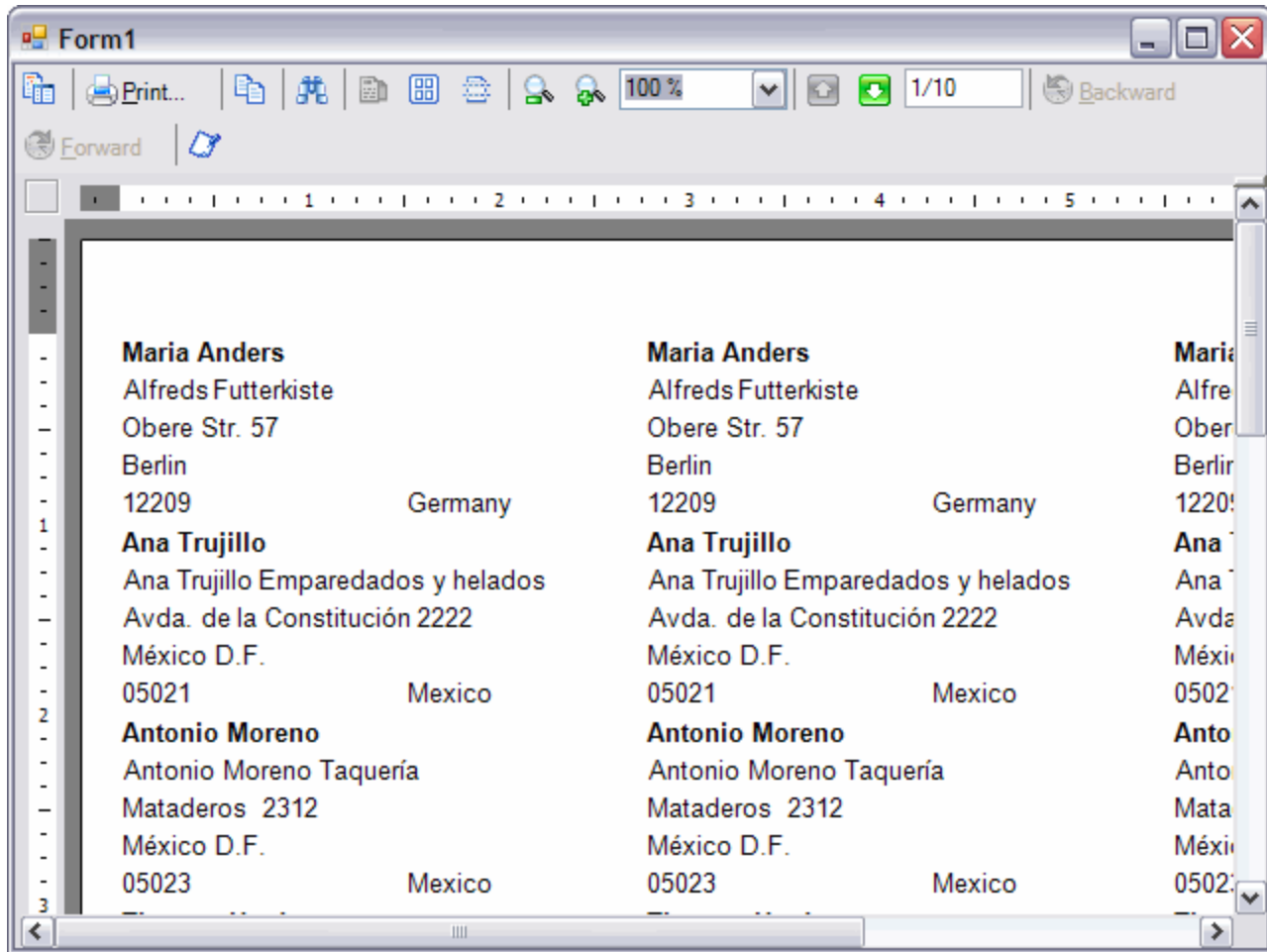


Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at `C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB` (on a **64-bit Windows operating system**, a copy is located in `C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB`).

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



To connect the report to a database

1. Add a report to a Visual Studio project, naming it **rptLabels**.
2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

PageHeader1

Detail1

3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, paste the following SQL query.

SQL Query

```
SELECT ContactName, CompanyName, Address, City, PostalCode, Country FROM Customers
```

8. Click **OK** to return to the report design surface.

To add controls to the report

1. Remove the PageHeader and Footer sections from the report by right-clicking in the PageHeader section and selecting **Delete**.
2. In the Report menu, select **Settings** and change the margins as follows:
 - Top margin: 0.5
 - Bottom margin: 0.5
 - Left margin: 0.2
 - Right margin: 0.2
3. Select rptLabels in the Properties Window and Set the **PrintWidth** property of the report to **8.1** (the width of the label sheet less the Left and Right margins).
4. Click on the detail section of the report to select it and make the following changes:
 - Set the **CanGrow** property to **False** (to maintain the label size)
 - Change the **ColumnCount** property to **3** (for three labels across the page)
 - Change the **ColumnDirection** property to **AcrossDown** (to have labels print in left-to-right order instead of top-to-bottom)
 - Set the **ColumnSpacing** property to **0.2** (to allow for blank space between labels)
 - Set the height of the Detail section to **1** (the height of the label, one inch)
5. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the detail section and set the **Size** and **Location** properties of each textbox as indicated.



Note: When you drag a field from the Report Explorer onto the design surface of the report, the **DataField**, **Name** and **Text** properties of the textbox object are automatically set to *txtFieldName1*.

Detail fields

| Field | Font | Size | Location |
|-------------|------------------|------------------|------------|
| ContactName | Font Bold = True | 2.5, 0.2 | 0, 0 |
| CompanyName | leave at default | 2.5, 0.2 | 0, 0.198 |
| Address | leave at default | 2.5, 0.2 | 0, 0.396 |
| City | leave at default | 2.5, 0.2 | 0, 0.594 |
| PostalCode | leave at default | 1.45, 0.2 | 0, 0.792 |
| Country | leave at default | leave at default | 1.5, 0.792 |

6. Select all of the textboxes, and in the Properties Window, set the **CanGrow** property to **False**. This prevents overlapping text, but may crop data if one of the fields contains more data than the control size allows.

If you preview the report at this point, one copy of each label appears on the page.

To add code to the detail_Format event to repeat labels

1. Double-click in the detail section to create a detail_Format event.
2. Add the following code to the event to repeat each label across all three columns.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
'print each label three times
Static counter As Integer
counter = counter + 1
```

```
If counter <= 2 Then
    Me.LayoutAction = LayoutAction.MoveLayout Or LayoutAction.PrintSection
Else
    Me.LayoutAction = LayoutAction.MoveLayout Or LayoutAction.NextRecord Or
LayoutAction.PrintSection
    counter = 0
End If
```

To write the code in C#

C# code. Paste JUST ABOVE the Format event.


```
int counter=0;
```

C# code. Paste INSIDE the Format event.

```
//print each label three times
counter = counter + 1;
if (counter <= 2)
{
    this.LayoutAction = LayoutAction.MoveLayout|LayoutAction.PrintSection;
}
else
{
    this.LayoutAction =
LayoutAction.MoveLayout|LayoutAction.NextRecord|LayoutAction.PrintSection;
    counter = 0;
}
```


Columnar Reports

ActiveReports supports newspaper column layouts 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 the Group section (i.e. start a new column on the change of a group). There is also a Boolean `ColumnGroupKeepTogether` property on the `GroupHeader`. When set to `True`, the `ColumnGroupKeepTogether` property attempts to prevent a group from splitting across columns. If a group cannot fit in the current column, it tries the next. If the group is too large for a single column, the property is ignored.

 **Note:** The `ColumnGroupKeepTogether` property is only implemented when the `GroupHeader`'s `GroupKeepTogether` property is set to `All`.

This walkthrough illustrates how to create a simple report using columns, and is split up into the following activities:

- Connecting the report to a data source
- Adding controls to the report to display data

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at `C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB` (on a **64-bit Windows operating system**, a copy is located in `C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB`).

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

| Alignment | Font | Text | Location | Size |
|------------------|-------------|------------------------------------|-----------------|--------------|
| Center | Arial, 14pt | Customer Telephone List by Country | 0, 0 | 6.5, 0.25 in |

- Select the Detail section, and make the following changes in the Properties Window:
 - Change the **CanShrink** property to **True** to eliminate white space after each name.
 - Change the **ColumnCount** property to **2** to split the detail section into two columns.
- Drag the following fields from the Report Explorer into the Detail section and set their properties as indicated:


| Field | Font | Location | Size |
|--------------|-------------|-----------------|--------------|
| CompanyName | Arial, 8pt | 0, 0 in | 1.15, 0.2 in |
| ContactName | Arial, 8pt | 1.16, 0 in | 1.15, 0.2 in |
| Phone | Arial, 8pt | 2.3, 0 in | 0.95, 0.2 in |

Overlaying Reports (Letterhead)

ActiveReports allows you to overlay static report formats over data reports. This walkthrough illustrates how to overlay an ActiveReport with a static letterhead report.

This walkthrough is split up into the following activities:

- Connecting the data report to a data source
- Adding controls to the letterhead and data reports
- Adding code to overlay the data report pages with the letterhead report

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).


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




| Customers in Argentina | | | |
|------------------------|----------------------------|--|--------------|
| ID | CompanyName | Address | City |
| RANCH | Rancho grande | Av. del Libertador 900 | Buenos Aires |
| OCEAN | Océano Atlántico Ltda. | Ing. Gustavo Moncada 8585 Piso 20-A | Buenos Aires |
| CACTU | Cactus Comidas para llevar | Cerrito 333 | Buenos Aires |
| Customers in Austria | | | |
| ID | CompanyName | Address | City |
| PICCO | Piccolo und mehr | Geislweg 14 | Salzburg |
| ERNSH | Ernst Handel | Kirchgasse 6 | Graz |
| Customers in Belgium | | | |
| ID | CompanyName | Address | City |
| MAISD | Maison Dewey | Rue Joseph-Bens 532 | Bruxelles |
| SUPRD | Suprêmes dâlices | Boulevard Tirou, 255 | Charleroi |
| Customers in Brazil | | | |
| ID | CompanyName | Address | City |

To connect rptData to a data source

1. Add two reports to a Visual Studio project, naming them **rptLetterhead** and **rptData**.
2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

 PageHeader1

 Detail1

3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Customers ORDER BY Country
```

8. Click **OK** to save the data source and return to the report design surface.

To add controls to rptData

1. Select the **PageHeader** section, and in the Properties Window, set the **Height** property to **0.65**. (This will match the height of the page header in the template.)

- Right-click the report and select **Insert > GroupHeader/Footer** to add group header and group footer sections.
- Make the following changes to the group header:
 - Change the **Name** property to **ghCustomers**
 - Change the **BackColor** property to **MediumSlateBlue**
 - Change the **CanShrink** property to **True**
 - Change the **DataField** property to **Country**
 - Change the **GroupKeepTogether** property to **FirstDetail**
 - Change the **KeepTogether** property to **True**
- Add the following controls to ghCustomers with properties set as indicated:
Group header labels

| Control | Miscellaneous | Text (or DataField) | Size | Location |
|---------|---|---|---------------|-------------|
| TextBox | Bold ForeColor = White Font Size = 12 | = "Customers in " + Country (DataField) | 2, 0.2 in | 0, 0 in |
| Label | Bold ForeColor = DarkSlateBlue | ID | 0.6, 0.198 in | 0, 0.2 in |
| Label | Bold ForeColor = DarkSlateBlue | Company Name | 1.1, 0.198 in | 0.7, 0.2 in |
| Label | Bold ForeColor = DarkSlateBlue | Address | 1, 0.198 in | 2.7, 0.2 in |
| Label | Bold ForeColor = DarkSlateBlue | City | 1, 0.198 in | 5.5, 0.2 in |

- Make the following changes to the detail section:
 - Change the **BackColor** property to **LightGray**
 - Change the **CanShrink** property to **True**
- In the Report Explorer, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the detail section and set the properties of each textbox as indicated.
Detail fields

| Field | Size | Location |
|-------------|-------------|-----------|
| CustomerID | 0.6, 0.2 in | 0, 0 in |
| CompanyName | 2, 0.2 in | 0.7, 0 in |
| Address | 2.8, 0.2 in | 2.7, 0 in |
| City | 1, 0.2 in | 5.5, 0 in |

- Change the group footer's **Height** property to **0**.

To add static controls to rptLetterhead

- Make the following changes to the page header:
 - Change the **BackColor** property to **DarkSlateBlue**
 - Change the **Height** property to **0.65**
- Add the following controls to the page header with properties set as indicated:
Page header labels

| Control | Miscellaneous | Size | Location |
|---------|---|--------------------|-----------|
| Label | Size = 36 Style = Bold ForeColor = White Text = GrapeCity | 3.7, 0.65 in | 0, 0 in |
| Picture | PictureAlignment = TopLeft Image (click ellipsis, navigate to <i>C:\Program Files\GrapeCity\ActiveReports 6\Introduction</i> (on a 64-bit Windows operating system , navigate to <i>C:\Program Files (x86)\GrapeCity\ActiveReports 6\Introduction</i>) and select itopimage1.gif) | 3, 0.65 in | 3.5, 0 in |

- Make the following changes to the page footer:
 - Change the **BackColor** property to **DarkSlateBlue**
- Add a label with the following properties to the page footer:
Page footer label

| Miscellaneous | ForeColor | Text | Size | Location |
|------------------------------------|-----------|---|----------------|----------|
| Alignment = Center Style = Bold | White | (919) 460-4551, http://www.grapecity.com , powersales@grapecity.com | 6.5, 0.2 in | 0, 0 in |

Adding code to overlay the data report pages with the letterhead report

To write the code in Visual Basic.NET

- Add the ActiveReports viewer control to the Windows Form. Then, double-click the top of the Windows Form to create an event-handling method for the form's Load event. Add code to the handler to:
 - Set the viewer to display the rptData report document
 - Overlay rptLetterhead on rptData

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

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Dim rpt As New rptData()
rpt.Run()
Viewer1.Document = rpt.Document
Dim rpt2 As New rptLetterhead()
rpt2.Run()
Dim i As Integer
For i = 0 To rpt.Document.Pages.Count - 1
    rpt.Document.Pages(i).Overlay(rpt2.Document.Pages(0))
Next
```

To write the code in C#

- Add the ActiveReports viewer control to the Windows Form. Then, double-click the top of the Windows Form to create an event-handling method for the form's Load event. Add code to the handler to:
 - Set the viewer to display the rptData report document
 - Overlay rptLetterhead on rptData

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

C# code. Paste INSIDE the Form Load event.

```
rptData rpt = new rptData();
rpt.Run();
viewer1.Document = rpt.Document;
rptLetterhead rpt2 = new rptLetterhead();
rpt2.Run();
for(int i = 0; i < rpt.Document.Pages.Count; i++)
{
    rpt.Document.Pages[i].Overlay(rpt2.Document.Pages[0]);
}
```

Chart Walkthroughs

Charts add quick visual impact to your reports, and allow data to be readily grasped even by casual readers. With a built-in chart control, ActiveReports makes it easy to provide premium reporting without the need to purchase extra tools.

Bar Chart

Describes how to create a bar chart which compares items across categories.

3D Pie Chart

Describes how to create a three dimensional pie chart which shows how the percentage of each data item contributes to a total percentage.

Financial Chart

Describes how to create a financial chart which lets you plot high, low, opening, and closing prices.

Simple Unbound Chart

Describes how to create a simple unbound chart.

Bar Chart

Bar charts are useful in comparing items across categories. This walkthrough illustrates how to create a simple bar chart using the ActiveReports chart control.

The walkthrough is split up into the following activities:

- Adding a chart control to the report
- Setting a data source for the chart
- Setting the chart's properties



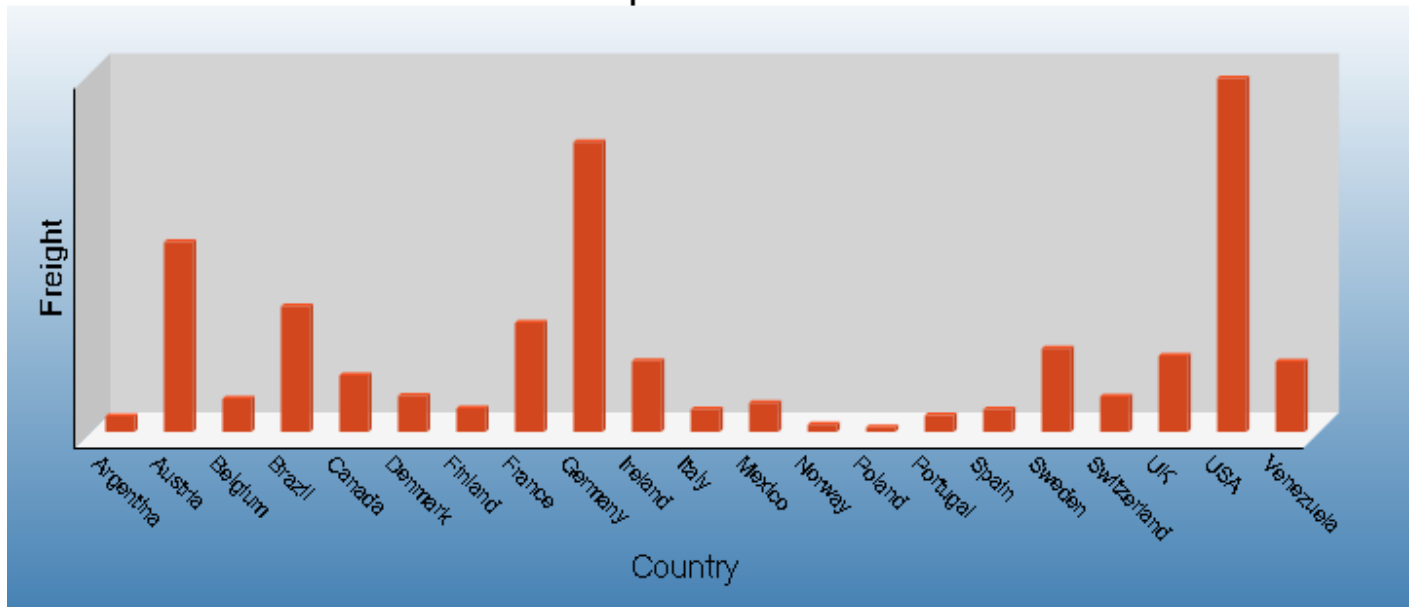
Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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

Simple Bar Chart



To add a chart control to the report

1. Resize the section in which you want to place the chart.
2. Click the ChartControl in the ActiveReports toolbox and draw it onto the report.
3. If the chart wizard appears, click **Cancel**.



Tip: If you do not want the chart wizard to appear each time you add a chart, clear the **Auto Run Wizard** checkbox. You can still access the wizard via the command verbs (see below).

4. Select the ChartControl and set its **Size** property to 7.5, 3.5 in.

To set a data source for the chart

1. With the chart control highlighted, click the Data Source verb below the Properties Window.



Tip: If the verb is not visible, right-click an empty space in the Properties Window and select **Commands** to display verbs.



2. In the Chart DataSource dialog box that appears, click the **Build** button.
3. In the Data Link Properties window, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis button (...) to browse to the Northwind database. Click **Open** once you have selected the file.
5. Click the **OK** button to close the window and fill in the Connection String.
6. In the Query field, enter the following SQL query

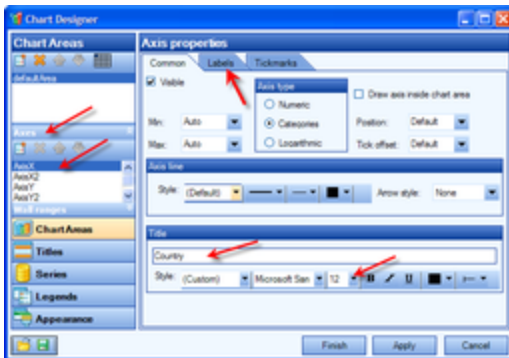
SQL Query

```
SELECT ShipCountry, SUM(Freight) AS FreightSum FROM Orders GROUP BY ShipCountry
```

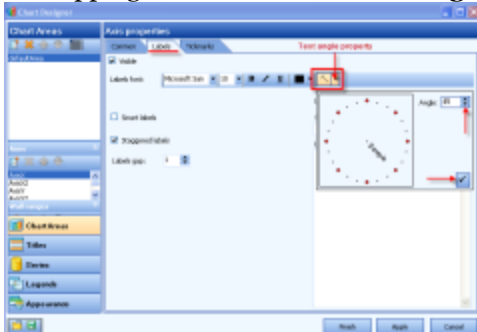
7. Click **OK** to save the data source and return to the report design surface.

To set the chart's properties

1. With the chart control highlighted, click the **Customize** verb below the Properties Window to open the Chart Designer window.
2. In the **ChartAreas** view which displays by default, click the **Axes** bar to expand it.
3. Click **Axis X**, and on the **Common** tab in the pane to the right, type **Country** in the **Title** textbox and increase the font size to **12**.



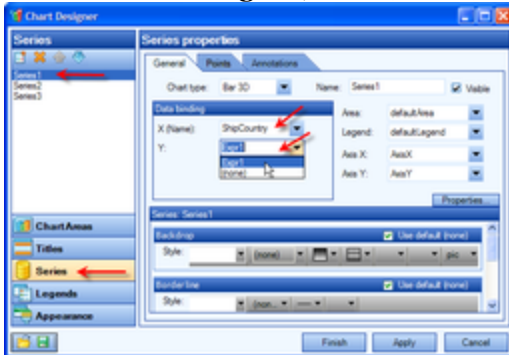
4. Choose the **Labels** tab in the Axis Properties pane on the right, select the **Staggered Labels** option to avoid overlapping labels and set the **Text angle** property to 45.



5. Click **Axis Y** on the left, and on the **Common** tab in the pane to the right, type **Freight** in the **Title** textbox and increase the font size to **12**.
6. Click the **Titles** bar on the left to expand it and display Title Properties in the pane to the right. In the list of titles, **header** is selected by default.
7. Type **Simple Bar Chart** in the **Caption** textbox, and increase the font size to **14**.



8. Select the **footer** in the list of titles to the left, and delete it.
9. Click the **Series** bar on the left to expand it and display Series Properties in the pane to the right. **Series1** is selected by default.
10. In the **Data Binding** box, set **X (Name)** to **ShipCountry**, and set **Y** to **FreightSum**.



11. Delete **Series2** and **Series3**.
12. Click the **Legend** bar on the left to expand it and display Legend Properties in the pane to the right. **defaultLegend** is selected by default.
13. Clear the **Visible** checkbox at the top of the **Common** tab to hide the legend.




14. Click the **Finish** button to exit the Chart Designer.

3D Pie Chart

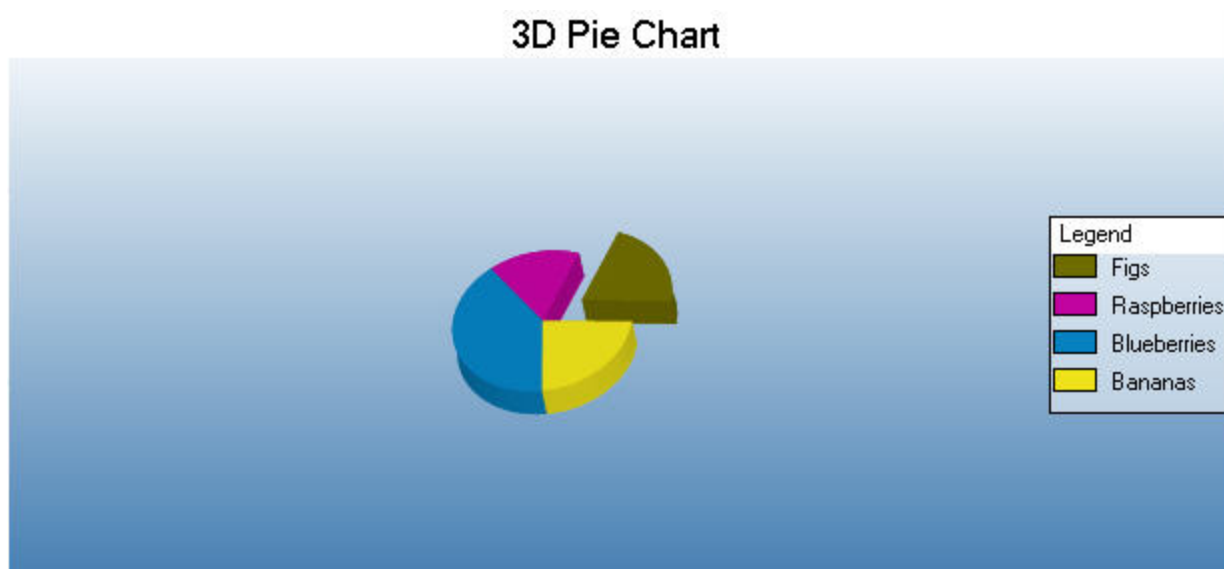
Pie charts are useful in showing how the percentage of each data item contributes to the total. This walkthrough illustrates how to create a three dimensional pie chart.

The walkthrough is split up into the following activities:

- Adding a chart control to the report
- Adding a series and data points to the chart
- Setting the chart's properties


 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

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



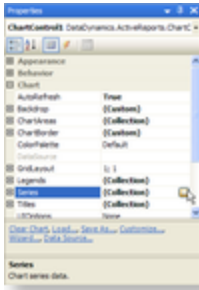
To add a chart control to the report

1. Resize the section in which you want to place the chart.
2. Click the ChartControl icon in the ActiveReports toolbox and draw it onto the report.
3. If the chart wizard appears, click **Cancel**.

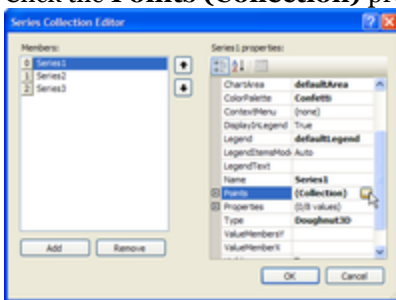
 **Tip:** If you do not want the chart wizard to appear each time you add a chart, clear the **Auto Run Wizard** checkbox. You can still access the wizard via the command verbs (see below).

To add a series and data points to the chart

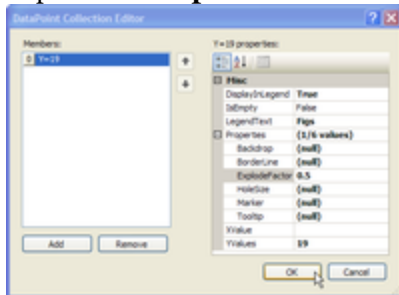
1. With the chart control selected, click the **Series (Collection)** property in the Properties Window, then click the ellipsis button (...) that appears.



2. In the Series Collection Editor window that appears, **Series1** is selected by default.
3. Under Series1 Properties, change the **ColorPalette** property to **Confetti**.
4. Change the **Type** property to **Doughnut3D**.
5. Click the **Points (Collection)** property, then click the ellipsis button that appears.



6. In the DataPoint Collection window that appears, click the **Add** button to add a data point
 - Set the **LabelText** property to **Figs**.
 - Set the **YValues** property to **19**.
 - Expand the **Properties** node and set the **ExplodeFactor** property to **.5** to pull this slice out from the pie.



7. Click the **Add** button to add another data point.
 - Set its **LabelText** property to **Raspberries**.
 - Set its **YValues** property to **15**.
8. Click the **Add** button to add another data point.
 - Set its **LabelText** property to **Blueberries**.
 - Set its **YValues** property to **37**.
9. Click the **Add** button to add another data point.
 - Set its **LabelText** property to **Bananas**.
 - Set its **YValues** property to **21**.
10. Click **OK** to save the data points and return to the Series Collection Editor.
11. Remove **Series2** and **Series3**.
12. Click **OK** to save the changes and return to the report design surface.

To set the chart's properties


1. With the chart control highlighted, click the **ChartAreas (Collection)** property in the Properties Window, then click the ellipsis button that appears.
2. In the ChartArea Collection Editor window that appears, expand the **Projection** property node and change the **VerticalRotation** property to **50**. This allows you to see more of the top of the pie.
3. Click **OK** to return to the report design surface.
4. With the chart control highlighted, click the **Titles (Collection)** property in the Properties Window, then click the ellipsis button that appears.
5. In the Title Collection Editor window that appears, under header properties, change the **Text** property to **3D Pie Chart**.
6. Expand the **Font** property node and set the **Size** to **14** to make your title stand out more.
7. Remove the footer title.
8. Click **OK** to return to the report design surface.

Financial Chart

Financial charts are useful for displaying stock information using High, Low, Open and Close values. This walkthrough illustrates how to create a Candle chart.

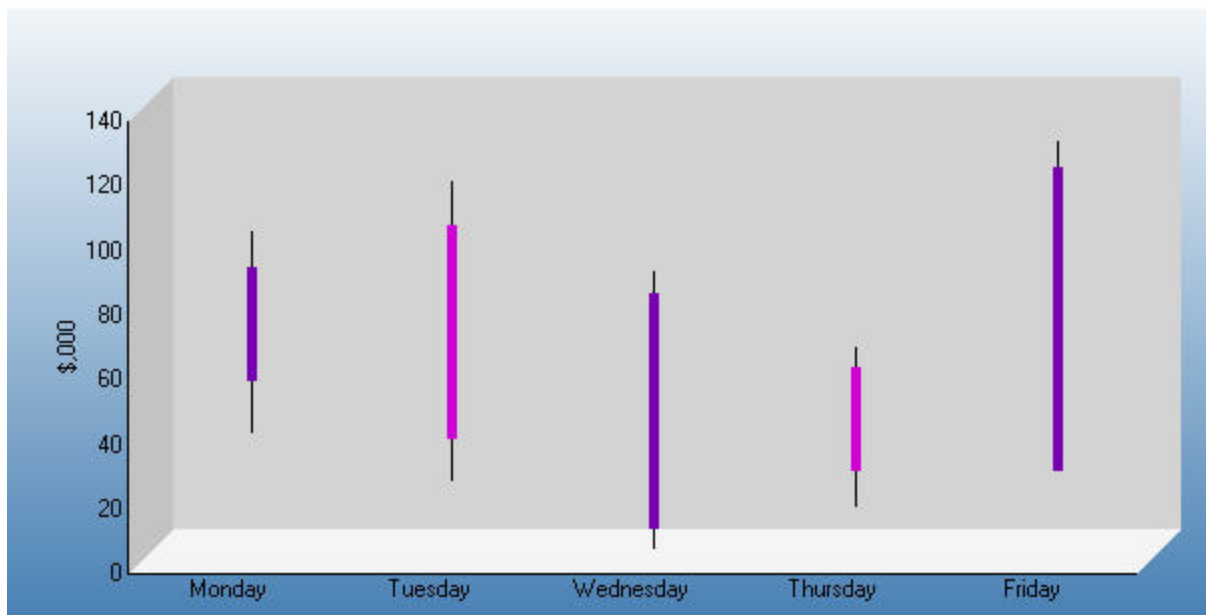
The walkthrough is split up into the following activities:

- Adding a chart control to the report
- Adding a series and data points to the chart
- Setting the chart's properties

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

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

Candle Chart

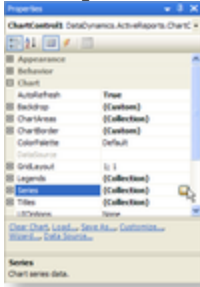


To add a chart control to the report

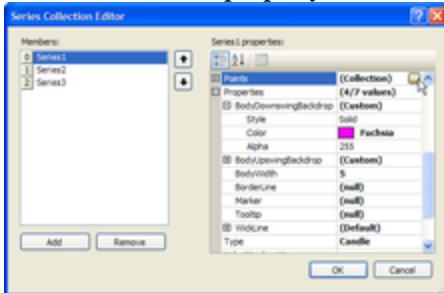
1. Resize the section in which you want to place the chart.
2. Click the ChartControl in the ActiveReports toolbox and draw it onto the report.

To add a series and data points to the chart

1. With the chart control highlighted, click the **Series** (Collection) property in the Properties Window, then click the ellipsis button that appears.



2. In the Series Collection Editor window that appears, **Series1** is selected by default.
3. Under Series1 Properties, change the **Type** property to **Candle**.
4. Expand the **Properties** node and set the **BodyDownswingBackdrop** property to **(Default)** so that you can expand the node and change the **Color** property to **Fuchsia**.
5. Set the **BodyUpswingBackdrop** property to **(Default)** so that you can expand the node and change the **Color** property to **DarkViolet**.
6. Set the **BodyWidth** property to **5** to make the opening to closing figure bar, the candle, wider than the wick.
7. Set the **WickLine** property to **(Default)** for a black wick.



8. Click the **Points** (Collection) property, then click the ellipsis button that appears.
9. In the DataPoint Collection window that appears, click **Add** to add a data point
 - Set its **YValues** property to **99; 37; 53; 88**.

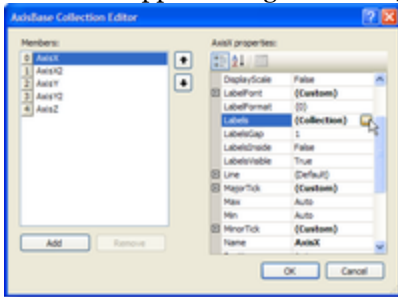
Note: The first Y value is the high figure or top of the wick; the second is the low figure, or bottom of the wick; the third is the opening figure; the fourth is the closing figure. If the fourth figure is higher than the third, the candle is DarkViolet, the BodyUpswingBackdrop.

- Add another data point, and set its **YValues** property to **115; 22; 101; 35**.
 - Add another data point, and set its **YValues** property to **87; 1; 7; 80**.
 - Add another data point, and set its **YValues** property to **63; 14; 57; 25**.
 - Add another data point, and set its **YValues** property to **130; 25; 25; 120**.
10. Click **OK** to save the data points and close the window.
 11. Back on the Series Collection Editor window, set the **Legend** property to **(none)**.
 12. Remove **Series2** and **Series3**.
 13. Click **OK** to return to the report design surface.

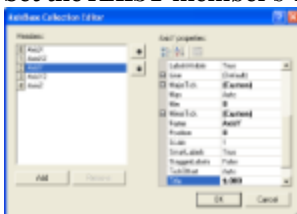
To set the chart's properties

Axes

1. With the chart control highlighted, click the **ChartAreas** (Collection) property in the Properties Window, then click the ellipsis button that appears.
2. In the ChartArea Collection Editor window that appears, under defaultArea properties, click the **Axes** (Collection) property, then click the ellipsis button that appears.
3. In the AxisBase Collection Editor window that appears, the AxisX member is selected by default. Under AxisX properties, delete the text from the **Title** property.
4. Click the **Labels** (Collection) property, then click the ellipsis button that appears. This is where you add the labels that appear along the X axis, the line across the bottom of the chart.



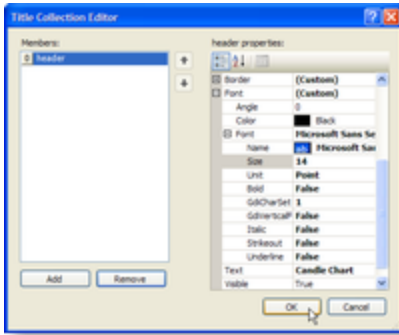
5. In the Array Data Editor window that appears, enter the following into the editor, each item on a separate line:
 - Monday
 - Tuesday
 - Wednesday
 - Thursday
 - Friday
6. Click the **OK** button to return to the AxisBase Collection Editor.
7. Select the **AxisY** member, and under AxisY properties, expand the **MajorTick** property node and set the **Step** property to **10**. This controls the numeric labels along the Y axis, the line along the left side of the chart.
8. Set the **LabelsVisible** property to **True**.
9. Set the **Min** property to **0**.
10. Set the **AxisY** member's **Title** property to **\$,000** and click **OK** to return to the ChartArea Collection Editor.



11. Click **OK** to return to the report design surface and see the changes reflected in the chart.

Title and Legend

1. With the chart control highlighted, click the **Titles** (Collection) property in the Properties Window, then click the ellipsis button that appears.
2. In the Title Collection Editor that appears, under header properties, change the **Text** property to **Candle Chart**.
3. Expand the **Font** property and set the Size to **14**.
4. Remove the footer title, and click **OK** to return to the report design surface.



5. With the Chart control highlighted, click the **Legends** (Collection) property in the Properties Window, then click the ellipsis button that appears.
6. In the Legend Collection Editor window that appears, set the **Visible** property to **False**, and click **OK** to return to the report design surface and see the completed chart.

Unbound Chart

The Chart control allows you to bind charts to any type of data source, including arrays. You can create a chart without setting its data source and load the data into the control at run time. This walkthrough illustrates how to create a simple unbound chart.

The walkthrough is split up into the following activities:

- Adding the chart control to the report and setting chart properties
- Adding code to create the chart at run time



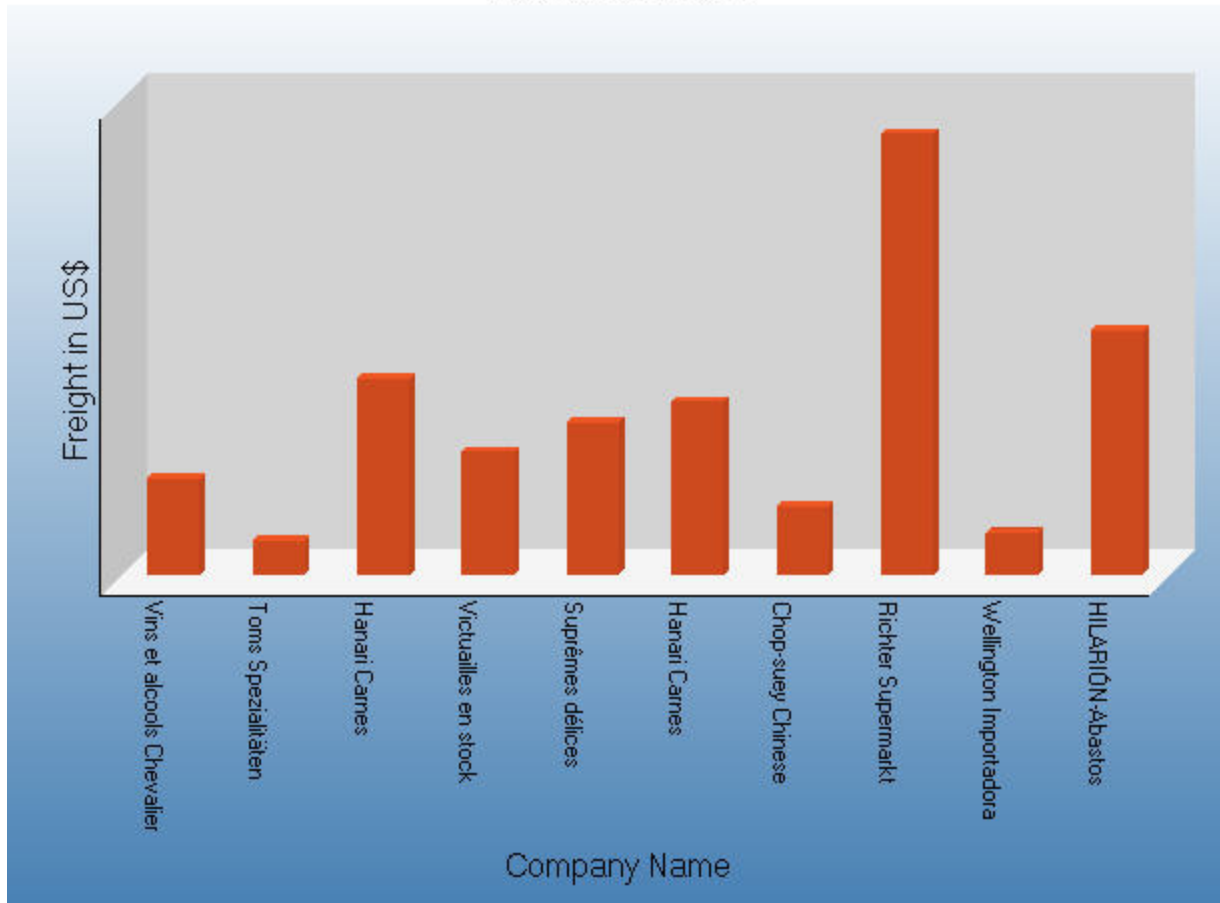
Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

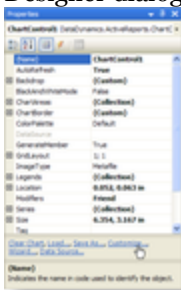
When you have finished this walkthrough, you will have a report that looks similar to the following.

Unbound Chart



To add a chart control to the report and set its properties

1. Resize the section in which you want to place the chart.
2. Click the ChartControl in the ActiveReports toolbox and draw it onto the report.
3. If the chart wizard appears, click **Cancel**.
4. Select the ChartControl and set its **Size** property to **6, 5** in.
5. With the chart control highlighted, click the **Customize** verb below the Properties Window to open the Chart Designer dialog.



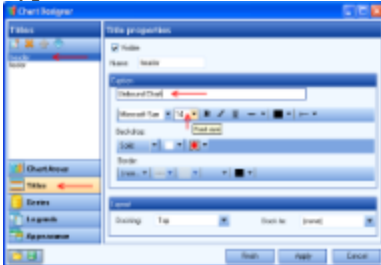
Tip: If you do not see the verbs, right-click an empty space in the Properties Window and select **Commands**.

6. In the **ChartAreas** view which displays by default, click the **Axes** bar to expand it.

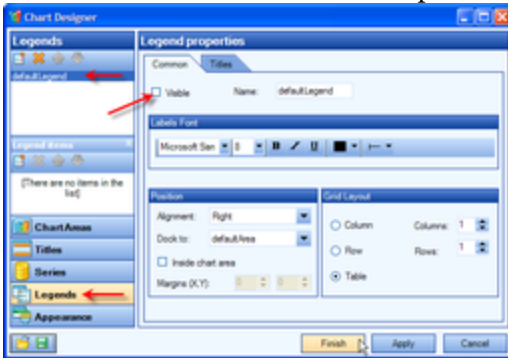
- Click **Axis X**, and on the **Common** tab in the pane to the right, type **Company Name** in the **Title** textbox and increase the font size to **12**.



- Click **Axis Y** on the left, and on the **Common** tab in the pane to the right, type **Freight in US\$** in the **Title** textbox and increase the font size to **12**.
- Click the **Titles** bar on the left to expand it and display Title Properties in the pane to the right. In the list of titles, **header** is selected by default.
- Type **Unbound Chart** in the **Caption** textbox, and increase the font size to **14**.



- Select the **footer** in the list of titles to the left, and delete it.
- Click the **Series** bar on the left to expand it and display Series Properties in the pane to the right.
- Delete **Series1**, **Series2** and **Series3**.
- Click the **Legends** bar on the left to expand it and display Legend Properties in the pane to the right. **defaultLegend** is selected by default.
- Clear the **Visible** checkbox at the top of the **Common** tab to hide the legend.




- Click the **Finish** button to exit the Chart Designer.

Back on the design surface of the report, the chart appears empty except for the title.

To write the code to create a chart at run time chart in Visual Basic or C#

Double-click in the gray area below rptUnbound. This creates an event-handling method for rptUnboundChart's ReportStart event. Add code to the handler to:

- Set the database path

 **Important:** Place this code above the ReportStart event.

- Create the series
- Create the dataset
- Set the chart properties
- Angle the labels to avoid overlap

The following examples show what the code for the methods look like in Visual Basic.NET and C#.

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
'Set the database path
Private Function getDatabasePath() As String
    Dim regKey As Microsoft.Win32.RegistryKey
    regKey = Microsoft.Win32.Registry.LocalMachine
    regKey = regKey.CreateSubKey("SOFTWARE\GrapeCity\ActiveReports 6\SampleDB")
    getDatabasePath = CType(regKey.GetValue(""), String)
End Function
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
'create the series
Dim series As New DataDynamics.ActiveReports.Chart.Series
series.Type = Chart.ChartType.Bar3D

'connection string and data adapter
Dim dbPath As String = getDatabasePath()
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + dbPath +
"\NWIND.mdb"
Dim da As New System.Data.OleDb.OleDbDataAdapter("SELECT * from Orders WHERE OrderDate
< #08/17/1994#", connString)

'create the dataset
Dim ds As New DataSet
da.Fill(ds, "Orders")

'set chart properties
Me.ChartControll1.DataSource = ds
Me.ChartControll1.Series.Add(series)
Me.ChartControll1.Series(0).ValueMembersY = ds.Tables("Orders").Columns(7).ColumnName
Me.ChartControll1.Series(0).ValueMemberX = ds.Tables("Orders").Columns(8).ColumnName

'angle the labels to avoid overlapping
Me.ChartControll1.ChartAreas(0).Axes(0).LabelFont.Angle = 90
```

C# code. Paste JUST ABOVE the ReportStart event.

```
//Set the database path
private string getDatabasePath()
{
    Microsoft.Win32.RegistryKey regKey = Microsoft.Win32.Registry.LocalMachine;
    regKey = regKey.CreateSubKey("SOFTWARE\GrapeCity\ActiveReports 6\SampleDB");
    return ((string) (regKey.GetValue("")));
}
```

C# code. Paste INSIDE the ReportStart event.

```
//create the series
DataDynamics.ActiveReports.Chart.Series series = new
DataDynamics.ActiveReports.Chart.Series();
series.Type = DataDynamics.ActiveReports.Chart.ChartType.Bar3D;

//connection string and data adapter
string dbPath = getDatabasePath();
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source= " + dbPath +
"\NWIND.mdb";
System.Data.OleDb.OleDbDataAdapter da = new System.Data.OleDb.OleDbDataAdapter
("SELECT * from Orders WHERE OrderDate < #08/17/1994#", connString);

// create the dataset
System.Data.DataSet ds = new System.Data.DataSet();
da.Fill(ds, "Orders");

// set chart properties
this.chartControl1.DataSource = ds;
this.chartControl1.Series.Add(series);
this.chartControl1.Series[0].ValueMembersY = ds.Tables["Orders"].Columns[7].ColumnName;
this.chartControl1.Series[0].ValueMemberX = ds.Tables["Orders"].Columns[8].ColumnName;

// angle the labels to avoid overlapping
this.chartControl1.ChartAreas[0].Axes[0].LabelFont.Angle = 90;
```

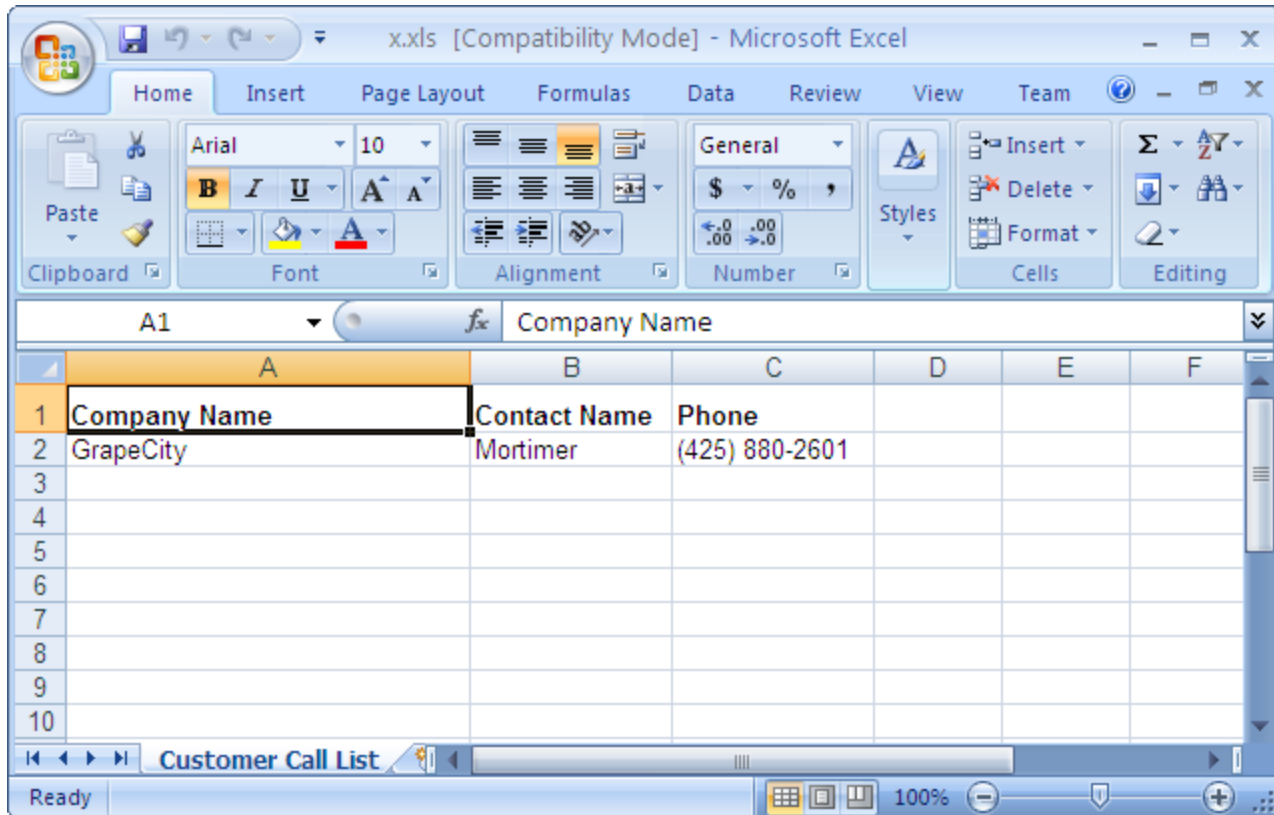
Basic Spreadsheet with SpreadBuilder

Included with the ActiveReports Excel export filter is the SpreadBuilder API. With this utility, you can create Excel spreadsheets cell by cell for maximum control. This walkthrough illustrates how to create a simple custom spreadsheet cell by cell, and save it to an Excel file.

This walkthrough is split up into the following activities:

- Adding the Excel export filter to your project
- Adding an ActiveReport.Document reference to the project
- Creating a Workbook using code
 - Adding a sheet to the Workbook's Sheets collection
 - Setting properties on columns and rows in the sheet
 - Setting values of cells in the sheet
 - Using the Save method to create an Excel file
- Viewing the Excel File

When you have completed this walkthrough, a custom Excel file like the following is created in the Bin/Debug subfolder of your project's folder.



To add the export filter to your project

1. Drag the XlsExport export control from the appropriate Visual Studio toolbox tab onto a Windows Form. (See **Adding ActiveReports Controls** for information if you haven't added the control to your toolbox.)
2. The control is visually represented in a component tray below the Windows Form and a reference to ActiveReports.XlsExport is added to your class references.

To add an ActiveReport.Document reference to your project

1. From the Visual Studio **Project** menu, select **Add New Item**.
2. In the Add New Item window that appears, select **ActiveReports 6 (code-based) File** and click the **Add** button. This adds an ActiveReport to the project.
3. You can delete the ActiveReport, leaving the ActiveReports.Document reference in your class references.

To write the code to create a workbook in Visual Basic or C#

Double-click the title bar of the Windows Form to create an event-handling method for the Form_Load event. Add code to the handler to:

- Create a Workbook, and add a sheet to the Workbook's Sheets collection
- Set properties on columns and rows in the sheet
- Set values of cells in the sheet
- Use the Save method to create an Excel file

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste inside the form Load event.

```
'Dimension a Workbook and add a sheet to its Sheets collection
Dim sb As New DataDynamics.SpreadBuilder.Workbook()
sb.Sheets.AddNew()

'Set up properties and values for columns, rows, and cells as desired
With sb.Sheets(0)
    .Name = "Customer Call List" 'sets the name of the sheet
    .Columns(0).Width = 2 * 1440 'sets the width of the 1st column
    .Columns(1).Width = 1440
    .Columns(2).Width = 1440
    .Rows(0).Height = 1440 / 4

    'Header row
    .Cell(0, 0).SetValue("Company Name")
    .Cell(0, 0).FontBold = True
    .Cell(0, 1).SetValue("Contact Name")
    .Cell(0, 1).FontBold = True
    .Cell(0, 2).SetValue("Phone")
    .Cell(0, 2).FontBold = True

    'First row of data
    .Cell(1, 0).SetValue("GrapeCity")
    .Cell(1, 1).SetValue("Mortimer")
    .Cell(1, 2).SetValue("(425) 880-2601")
End With

'Save the Workbook to an Excel file
sb.Save(Application.StartupPath & "\x.xls")
MessageBox.Show("Your Spreadsheet has been saved to " & Application.StartupPath &
"\x.xls")
```

To write the code in C#**C# code. Paste inside the form Load event.**

```
//Dimension a Workbook and add a sheet to its Sheets collection
DataDynamics.SpreadBuilder.Workbook sb = new DataDynamics.SpreadBuilder.Workbook();
sb.Sheets.AddNew();

//Set up properties and values for columns, rows and cells as desired
sb.Sheets[0].Name = "Customer Call List";
sb.Sheets[0].Columns(0).Width = 2 * 1440;
sb.Sheets[0].Columns(1).Width = 1440;
sb.Sheets[0].Columns(2).Width = 1440;
sb.Sheets[0].Rows(0).Height = 1440/4;

//Header row
sb.Sheets[0].Cell(0,0).SetValue("Company Name");
sb.Sheets[0].Cell(0,0).FontBold = true;
sb.Sheets[0].Cell(0,1).SetValue("Contact Name");
sb.Sheets[0].Cell(0,1).FontBold = true;
sb.Sheets[0].Cell(0,2).SetValue("Phone");
sb.Sheets[0].Cell(0,2).FontBold = true;

//First row of data
sb.Sheets[0].Cell(1,0).SetValue("GrapeCity");
```

```
sb.Sheets[0].Cell(1,1).SetValue("Mortimer");
sb.Sheets[0].Cell(1,2).SetValue("(425) 880-2601");

//Save the Workbook to an Excel file
sb.Save(Application.StartupPath + @"\x.xls");
MessageBox.Show("Your Spreadsheet has been saved to " + Application.StartupPath +
@"\x.xls");
```

To view the Excel File

1. Press **F5** to run the project. A message box informs you of the exact location of the exported x.xls file.
2. Navigate to the Bin/Debug subfolder of your project's folder and open the XLS file.

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 to retrieve the grouping data from the database in the same manner as a textbox's DataField property. This walkthrough illustrates how to set up grouping in an unbound report.

This walkthrough is split into the following activities:

- 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
- Adding code to close the connection to the data source



Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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

The screenshot shows a report with two sections: 'Beverages' and 'Condiments'. Each section has a table with 'Product Name' and 'Units in Stock' columns. The 'Beverages' table includes items like Chamomile Tea, Orange Juice, Sweetened Condensed Milk, etc. The 'Condiments' table includes items like Sweetener, Vanilla, etc. A summary row at the bottom of the 'Beverages' section shows 'Total Number of Beverages' with a value of 12.

| Product Name | Units in Stock |
|---------------------------|----------------|
| Chamomile Tea | 65 |
| Orange Juice | 17 |
| Sweetened Condensed Milk | 30 |
| Sweetened Condensed Milk | 111 |
| Strawberry Citrus | 30 |
| Tea | 39 |
| Tea de Menta | 17 |
| Tea de Menta | 17 |
| Laughing Lumberjack Lager | 52 |
| Outback Lager | 15 |
| Whispering Willows | 126 |
| Whispering Willows | 57 |
| Total Number of Beverages | |
| | 12 |

| Product Name | Units in Stock |
|----------------------------------|----------------|
| Sweetener | 36 |
| Vanilla | 6 |
| Original Frankfurter gummy Bells | 32 |
| Sweden's Representative Spread | 120 |
| Gula Melacca | 27 |

To add code to connect the report to a data source

1. Double-click the gray area below the report. This creates an event-handling method for the report's **ReportStart** event.
2. Add code to the handler to:

- Set the data source connection string
- Set the data source SQL query
- Open the connection and retrieve the data with the data reader

The following examples show what the code for the method looks like in Visual Basic.NET and C#.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim connection As System.Data.OleDb.OleDbConnection
Dim reader As System.Data.OleDb.OleDbDataReader
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
'Create the data connection and change the data source path as necessary
Dim connectionString As String
connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB"
connection = New System.Data.OleDb.OleDbConnection(connectionString)
connection.Open()

Dim sqlString As String
sqlString = "SELECT * FROM categories INNER JOIN products ON categories.categoryid=
products.categoryid ORDER BY categories.CategoryID"
Dim command As New System.Data.OleDb.OleDbCommand(sqlString, connection)
'Retrieve data
reader = command.ExecuteReader()
```

To write the code in C#

C# code. Paste JUST ABOVE the ReportStart event.

```
private System.Data.OleDb.OleDbConnection connection;
private System.Data.OleDb.OleDbDataReader reader;
```

C# code. Paste INSIDE the ReportStart event.

```
//Create the data connection and change the data source path as necessary
string connectionString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB";
connection=new System.Data.OleDb.OleDbConnection(connectionString);
connection.Open();

string sqlString = "SELECT * FROM categories INNER JOIN products ON
categories.categoryid = products.categoryid ORDER BY categories.CategoryID";
System.Data.OleDb.OleDbCommand command = new System.Data.OleDb.OleDbCommand(sqlString,
connection);

//Retrieve data
reader = command.ExecuteReader();
```

To add controls to the report to contain data

1. Back on the design surface of the report, right-click and select **Insert**, then **Group Header/Footer** to add group header and footer sections.
2. Select the group header and make the following changes in the Properties Window.
 - **Name:** ghCategories

- **BackColor:** Silver
 - **CanShrink:** True
 - **DataField:** CategoryID
 - **GroupKeepTogether:** All
 - **KeepTogether:** True
3. Select the group footer, and in the Properties Window, change the **Name** property to **gfCategories**.
 4. Select the detail section, and in the Properties Window, change the **CanShrink** property to **True**.
 5. Add the following controls to the GroupHeader section (drag the bottom edge of the section down to display all of the controls):

GroupHeader controls

| Control | DataField | Name | Text | Miscellaneous | Location |
|---------|--------------|-----------------|----------------|--|-------------|
| TextBox | CategoryName | txtCategoryName | Category Name | ForeColor = Blue BackColor = Silver Font size = 12 Size = 2, 0.2 in | 0, 0 in |
| TextBox | Description | txtDescription | Description | Size = 6, 0.198 in | 0, 0.3 in |
| Label | | lblProductName | Product Name | Bold | 0, 0.6 in |
| Label | | lblUnitsInStock | Units In Stock | Bold Alignment = Right | 4.4, 0.6 in |

6. Add the following controls to the Detail section:

Detail controls

| Control | DataField | Name | Text | Miscellaneous | Location |
|---------|--------------|-----------------|----------------|--------------------|-----------|
| TextBox | ProductName | txtProductName | Product Name | Size = 4, 0.198 in | 0, 0 in |
| TextBox | UnitsInStock | txtUnitsInStock | Units In Stock | Alignment = Right | 4.4, 0 in |


7. Add the following controls to the GroupFooter section:

GroupFooter controls

| Control | DataField | Name | Text | Miscellaneous | Location |
|---------|-------------|---------------|-------------|---|--|
| Label | TotalLabel | lblTotalLabel | | Size = 2.4, 0.198 in | 2, 0 in |
| TextBox | ProductName | txtTotalItems | Total Items | SummaryType = SubTotal SummaryFunc = Count SummaryRunning = Group SummaryGroup = ghCategories Alignment = Right | 4.4, 0 in |
| Line | | Line1 | | LineWeight = 3 | X1 = 1.88 in X2 = 6.44 in Y1 = 0 in Y2 = 0 in |

8. Right-click the **PageHeader** section and select **Delete**.

To add fields using the DataInitialize event

 **Warning:** Do not access the Fields collection outside the **DataInitialize** and **FetchData** events. Accessing the Fields collection outside of these events is not supported, and has unpredictable results.

To write the code in Visual Basic

1. Right-click in any section of the design surface of the report, and select **View Code** to display the code view for the report.
2. At the top left of the code view of the report, click the drop-down arrow and select **(YourReportName Events)**.
3. At the top right of the code window, click the drop-down arrow and select **DataInitialize**. This creates an event-handling method for the report's DataInitialize event.
4. 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.NET code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryID")
Fields.Add("CategoryName")
Fields.Add("ProductName")
Fields.Add("UnitsInStock")
Fields.Add("Description")
Fields.Add("TotalLabel")
```

To write the code in C#

1. Click in the gray area below the report to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **DataInitialize**. This creates an event-handling method for the report's DataInitialize event.
4. 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.

C# code. Paste INSIDE the DataInitialize event.

```
Fields.Add("CategoryID");
Fields.Add("CategoryName");
Fields.Add("ProductName");
Fields.Add("UnitsInStock");
Fields.Add("Description");
Fields.Add("TotalLabel");
```

To populate the fields using the FetchData event

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(YourReportName Events)**.
2. At the top right of the code window, click the drop-down arrow and select **FetchData**. This creates an event-handling method for the report's FetchData event.
3. 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.NET code. Paste INSIDE the FetchData event.

Try

```
reader.Read()
Me.Fields("CategoryID").Value = reader("categories.CategoryID")
Me.Fields("CategoryName").Value = reader("CategoryName")
Me.Fields("ProductName").Value = reader("ProductName")
Me.Fields("UnitsInStock").Value = reader("UnitsInStock")
Me.Fields("Description").Value = reader("Description")
Me.Fields("TotalLabel").Value = "Total Number of " + reader("CategoryName") + ":"
eArgs.EOF = False
Catch
    eArgs.EOF = True
End Try
```

To write the code in C#

1. Back in design view, click in the gray area below the report to select it.
2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
4. 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.

C# code. Paste INSIDE the FetchData event.

```
try
{
    reader.Read();
    Fields["CategoryID"].Value = reader["categories.CategoryID"].ToString();
    Fields["CategoryName"].Value = reader["CategoryName"].ToString();
    Fields["ProductName"].Value = reader["ProductName"].ToString();
    Fields["UnitsInStock"].Value = reader["UnitsInStock"].ToString();
    Fields["Description"].Value = reader["Description"].ToString();
    Fields["TotalLabel"].Value = "Total Number of " +
reader["CategoryName"].ToString() + ":";
    eArgs.EOF = false;
}
catch
{
    eArgs.EOF = true;
}
```

Adding code to close the connection to the data source

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select (**YourReportName Events**).
2. At the top right of the code window, click the drop-down arrow and select **ReportEnd**. This creates an event-handling method for the report's ReportEnd event.
3. Add code to the handler to close the connection.

Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
reader.Close()
connection.Close()
```

To write the code in C#

1. Back in design view, click in the gray area below the report to select it.

2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **ReportEnd**. This creates an event-handling method for the report's ReportEnd event.
4. Add code to the handler to close the connection.

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

C# code. Paste INSIDE the ReportEnd event.

```
reader.Close();  
connection.Close();
```

Subreport Walkthroughs

Follow step-by-step tutorials as you create Visual Studio projects using the ActiveReports subreport control. You can use the subreport control to embed a report within another report. Subreports are executed each time the parent section (i.e. the section in which the Subreport control is placed) is printed. Some ways to use subreports include:

- Repeating a group of relational data (for example, a list of orders in the main report, with ordered items in the subreport)
- Using multiple data sources within a report
- Creating multiple detail sections within a report

This section contains information about how to:

Subreports with Run-Time Data Sources

Learn how to embed a subreport in a main report, passing the data source from the main report to the subreport at run time.

Nested Subreports

Learn how to nest subreports to display main, child, and grandchild levels in a report.

Subreports with XML Data

Learn how to use XML data with subreports.

Subreports with Run-Time Data Sources

ActiveReports allows reports to contain any number of child reports 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 processed. This walkthrough illustrates how to modify the subreport record source from the data in the parent report to retrieve the correct information.

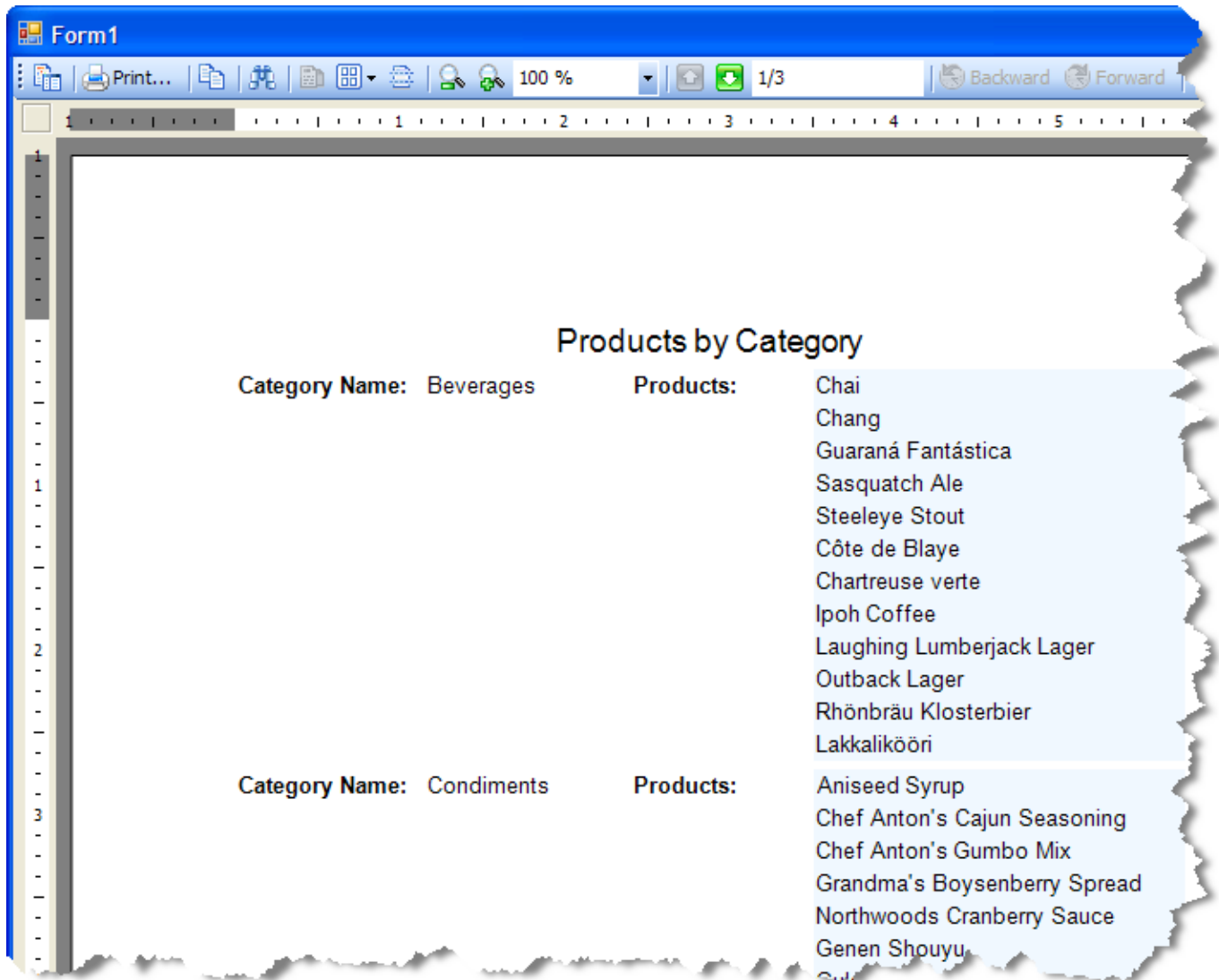
This walkthrough is split up into the following activities:

- Adding a main report and a subreport to a Visual Studio project
- Connecting the main report to a data source
- Adding controls to the main report to display data and contain the subreport
- Adding controls to the subreport to display data
- Adding code to save the current record's CategoryID for use in the subreport's SQL query
- Adding code to create an instance of the subreport
- Adding code to assign a data source for the subreport

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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

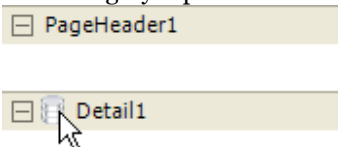


To add a main report and a subreport to a Visual Studio project

1. Open a new project in Visual Studio.
2. From the **Project** menu, select **Add New Item**.
3. Select **ActiveReports 6 (code-based) File** and rename the file **rptMain**.
4. Click **Add**.
5. Add a second report named **rptSub**.

To connect the main report to a data source

1. Click the gray report DataSource icon on the Detail section band of rptMain to open the Report Data Source dialog.



2. On the "OLE DB" tab, next to Connection String, click the **Build** button.

3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Categories
```

7. Click **OK** to save the data source and return to the report design surface.

To add controls to the main report to display data and contain the subreport

1. Click in the gray area below rptMain to select the report.
2. In the Properties Window, set the **PrintWidth** property to **5.75**.
3. Set the **CanShrink** property of the Detail section of rptMain to **True** to eliminate white space.
4. Drag a Label control onto the Page Header section of rptMain, setting the properties as indicated:
Label Properties


| Size | Font Size | Name | Miscellaneous | Text | Location |
|---------------|-----------|-----------------------|----------------|----------------------|----------|
| 5.75, 0.25 in | 14 | lblProductsbyCategory | Align = Center | Products by Category | 0, 0 in |

5. Drag the following controls onto the Detail section of rptMain, setting the properties as indicated:
Detail section controls

| Control | Miscellaneous | Name | Text | Location |
|-----------|--|------------------|----------------|---------------|
| Label | Bold Size = 1.15, 0.2 in | lblCategoryName | Category Name: | 0, 0.05 in |
| TextBox | DataField = CategoryName | txtCategoryName1 | | 1.15, 0.05 in |
| Label | Bold | lblProducts | Products: | 2.4, 0.05 in |
| SubReport | Size = 2.25, 1 in | SubReport1 | | 3.5, 0.05 in |
| TextBox | DataField = CategoryID Visible = False | txtCategoryID1 | | |

To add controls to the subreport to display data

1. Set the **CanShrink** property of the Detail section of rptSub to **True** to eliminate white space.
2. Set the **BackColor** property of the Detail section to **AliceBlue** to distinguish the subreport from the main report.

 **Tip:** Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

3. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
4. Add a TextBox control to the Detail section of rptSub, setting the properties as indicated:

TextBox properties

| Size | DataField | Name | Text | Location |
|------|-----------|------|------|----------|
|------|-----------|------|------|----------|

2.25, 0.198 in


ProductName

txtProductName

Product Name

0, 0 in

Adding code to create an instance of the subreport

 **Warning:** Do not create a new instance of the subreport in the **Format** event. Doing so creates a new subreport each time the section Format code is run, which uses a lot of memory.

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select (**rptMain Events**).
2. 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.
3. Add code to the handler to create a new instance of the subreport.

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

Visual Basic.NET code. Paste **JUST ABOVE** the ReportStart event.

```
Private rpt As rptSub
Private childDataSource As New DataDynamics.ActiveReports.DataSources.OleDbDataSource()
```

Visual Basic.NET code. Paste **INSIDE** the ReportStart event.

```
rpt = New rptSub()
```

To write the code in C#

1. Click in the gray area below rptMain to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportStart**. This creates an event-handling method for the report's ReportStart event.
4. Add code to the handler to create a new instance of the subreport.

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

C# code. Paste **JUST ABOVE** the ReportStart event.

```
private rptSub rpt;
private DataDynamics.ActiveReports.DataSources.OleDbDataSource childDataSource = new
DataDynamics.ActiveReports.DataSources.OleDbDataSource();
```

C# code. Paste **INSIDE** the ReportStart event.

```
rpt = new rptSub();
```

Adding code to assign a data source for the subreport

1. Back in design view of the report, double-click the detail section. This creates the Detail_Format event handler.
2. Add code to the handler to:
 - Set the connection string for the OleDbDataSource for the subreport
 - Set the SQL query for the new data source and pass in the current record's CategoryID
 - Set the data source of the subreport to the data source
 - Assign rptSub to the SubReport control

To write the code in Visual Basic

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

Visual Basic.NET code. Paste **INSIDE** the Format event.

```
childDataSource.ConnectionString = CType(Me.DataSource,
```

```
DataDynamics.ActiveReports.DataSources.OleDbDataSource).ConnectionString  
childDataSource.SQL = "SELECT * FROM Products WHERE CategoryID = " +  
Me.txtCategoryID1.Value.ToString  
rpt.DataSource = childDataSource  
Me.SubReport1.Report = rpt
```

To write the code in C#

C# code. Paste INSIDE the Format event.

```
childDataSource.ConnectionString =  
((DataDynamics.ActiveReports.DataSources.OleDbDataSource)this.DataSource).ConnectionString;  
childDataSource.SQL = "SELECT * FROM Products WHERE CategoryID = " +  
this.txtCategoryID1.Value.ToString();  
rpt.DataSource = childDataSource;  
this.SubReport1.Report = rpt;
```

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. This walkthrough illustrates how to set up embedded subreports.

This walkthrough is split up into the following activities:

- Creating parent, child, and grandparent reports
- Connecting each report to a data source
- Adding controls to each report to display the data
- Adding code to display reports in the subreport controls



Tip: For basic steps like viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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

| Customer Orders by Employee | | | |
|-----------------------------|-----------|--------------|-------------|
| Employee ID | Last Name | First Name | Extension |
| 1 | Davolio | Nancy | 5467 |
| Company Name | | Contact Name | Phone |
| Alfreds Futterkiste | | Maria Anders | 030-0074321 |
| Order ID | | Order Date | |
| 10249 | | 8/5/94 | |
| 10264 | | 8/24/94 | |
| 10271 | | 9/1/94 | |
| 10272 | | 9/2/94 | |
| 10274 | | 9/6/94 | |
| 10291 | | 9/27/94 | |

To create parent, child, and grandchild reports

1. Open a new project in Visual Studio.
2. From the **Project** menu, select **Add New Item**.
3. Select **ActiveReports 6 (code-based) File** and rename the file **rptEmployees**.
4. Click **Open**.
5. Repeat for **rptCustomers** and **rptOrders**.

To connect the parent report, rptEmployees, to a data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

PageHeader1

Detail1


2. On the "OLE DB" tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

```
SELECT Employees.EmployeeID, Employees.LastName, Employees.FirstName,
Employees.Extension, Customers.CustomerID
FROM Customers, Employees
```

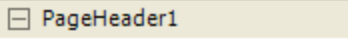
```
ORDER BY Employees.EmployeeID, Customers.CustomerID
```

- Click **OK** to save the data source and return to the report design surface.

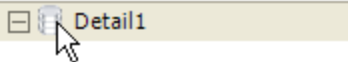
 **Note:** This query joins the Employees table for the parent report to the Customers table for the subreport.

To connect the child report, rptCustomers, to a data source

- Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1



Detail1

- On the "OLE DB" tab, next to Connection String, click the **Build** button.
- In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
- Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
- Click **OK** to close the window and fill in the Connection String field.
- In the Query field, enter the following SQL query

SQL Query

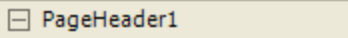
```
SELECT Customers.*, Employees.EmployeeID, Orders.OrderID
FROM Employees INNER JOIN (Customers INNER JOIN Orders ON Customers.CustomerID =
Orders.CustomerID) ON Employees.EmployeeID = Orders.EmployeeID
WHERE Customers.CustomerID = '<%CustomerID%>'
```

- Click **OK** to save the data source and return to the report design surface.

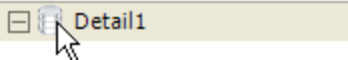
 **Note:** This SQL query uses parameters syntax: '<%CustomerID%>'. For more information on parameters, see the [Parameters](#) topic.

To connect the grandchild report, rptOrders, to a data source

- Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1



Detail1

- On the "OLE DB" tab, next to Connection String, click the **Build** button.
- In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
- Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
- Click **OK** to close the window and fill in the Connection String field.
- In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Orders WHERE EmployeeID = <%EmployeeID%>
```

- Click **OK** to save the data source and return to the report design surface.

To add controls to the parent report, rptEmployees, to display data

- Right-click on the design surface of **rptEmployees** and select **Insert**, then **Group Header/Footer** to add a group header and a group footer section to the report.
- With the **group header** selected in the Properties Window, make the following changes:
 - Name:** ghEmployees
 - DataField:** EmployeeID
 - Height:** 0.6
- Set the **Height** property of the page header section to **0.3**.
- Set the **CanShrink** property of the Detail section to **True** to eliminate white space.
- In the **Report Explorer**, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the group header section and set the properties of each textbox as indicated.

Group header fields

| DataField | Name | Location | Size |
|------------|----------------|--------------|--------------|
| EmployeeID | txtEmployeeID1 | 0, 0.3 in | 1, 0.2 in |
| LastName | txtLastName1 | 1.05, 0.3 in | 1.35, 0.2 in |
| FirstName | txtFirstName1 | 2.5, 0.3 in | 1.3, 0.2 in |
| Extension | txtExtension1 | 3.85, 0.3 in | 1, 0.2 in |

- Drag the following controls from the ActiveReports Toolbox onto the indicated section of rptEmployees, setting the properties as indicated.

Other controls

| Control | Section | Name | Text | Miscellaneous | Location | Size |
|-----------|-------------|------------|-----------------------------|--------------------------------------|------------|--------------|
| Label | PageHeader | label1 | Customer Orders by Employee | Font size = 14 Alignment = Center | 0, 0 in | 6.5, 0.25 in |
| Label | GroupHeader | label2 | Employee ID | Bold | 0, 0 in | 1, 0.198 in |
| Label | GroupHeader | label3 | Last Name | Bold | 1.05, 0 in | 1, 0.198 in |
| Label | GroupHeader | label4 | First Name | Bold | 2.5, 0 in | 1, 0.198 in |
| Label | GroupHeader | label5 | Extension | Bold | 3.85, 0 in | 1, 0.198 in |
| Subreport | Detail | subReport1 | | | 1.05, 0 in | 5.4, 1 in |

To add controls to the child report, rptCustomers, to display data

Since the subreport control in the parent report that displays this report is 5.4 inches wide, none of the controls on this report extend beyond 5.4 inches. If you need a visual reminder, you can resize the child report by dragging the right edge inward or by changing the PrintWidth property.

1. Click in the gray area below **rptCustomers** to select the report.
2. In the Properties Window, change the **ShowParametersUI** property to **False**. This prevents the subreport from requesting parameter values from the user.
3. Select the Detail section and set the **CanShrink** property to **True** to eliminate white space.
4. Set the **BackColor** property of the Detail section to **AliceBlue** to distinguish the subreport from the main report.



Tip: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

5. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
6. Drag the following controls from the ActiveReports toolbox onto the Detail section of rptCustomers, setting the properties as indicated.

Detail section controls

| Control | Name | DataField | Text | Miscellaneous | Location | Size |
|-----------|------------|-------------|--------------|---------------------------|---------------|-----------------|
| Label | label1 | | Company Name | Bold | 0, 0 in | 1, 0.198 in |
| Label | label2 | | Contact Name | Bold | 2, 0 in | 1, 0.198 in |
| Label | label3 | | Phone | Bold | 4.2, 0 in | 1, 0.198 in |
| TextBox | textBox1 | CompanyName | | | 0, 0.29 in | 1.9, 0.198 in |
| TextBox | textBox2 | ContactName | | | 2, 0.29 in | 2.125, 0.198 in |
| TextBox | textBox3 | Phone | | | 4.2, 0.29 in | 1, 0.198 in |
| Label | label4 | | Order ID | Bold | 2, 0.604 in | 1, 0.198 in |
| Label | label5 | | Order Date | Bold Alignment = Right | 3.3, 0.604 in | 1, 0.198 in |
| SubReport | subReport1 | | | | 2, 0.802 in | 3, 1 in |

To add controls to the grandchild report, rptOrders, to display data


Since the subreport control in the child report that displays this report is 3 inches wide, none of the controls on this report extend beyond 3 inches. If you need a visual reminder, you can resize the child report by dragging the right edge inward or by changing the PrintWidth property.

1. Click in the gray area below **rptOrders** to select the report.
2. In the Properties Window, change the **ShowParametersUI** property to **False**.
3. Select the Detail section and set the **CanShrink** property to **True** to eliminate white space.
4. Set the **BackColor** property of the Detail section to **Lavender** to distinguish the subreport from the main report.
5. Right-click the PageHeader or PageFooter section and select **Delete**.
6. Add the following controls to the Detail section of rptOrders, setting the properties as indicated.

Detail section controls

| Control | Name | DataField | Miscellaneous | Location | Size |
|---------|--------------|-----------|---|-----------|-------------|
| TextBox | txtOrderID | OrderID | | 0, 0 in | 1, 0.198 in |
| TextBox | txtOrderDate | OrderDate | OutputFormat = M/d/yy Alignment = Right | 1.3, 0 in | 1, 0.198 in |

To add code to display rptCustomers in the subreport control on rptEmployees

 **Warning:** Do not create a new instance of the subreport in the **Format** event. Doing so creates a new subreport each time the section Format code is run, which uses a lot of memory.

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(rptEmployees Events)**.
2. 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.
3. Add code to the handler to create a new instance of the subreport.

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

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Me.subReport1.Report = New rptCustomers()
```

To write the code in C#

1. Click in the gray area below rptEmployees to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportStart**. This creates an event-handling method for the report's ReportStart event.
4. Add code to the handler to create a new instance of the subreport.

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

C# code. Paste INSIDE the ReportStart event.

```
this.subReport1.Report = new rptCustomers();
```

To add code to display rptOrders in the subreport control on rptCustomers

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(rptCustomers Events)**.
2. 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.

3. Add code to the handler to create a new instance of the subreport.

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

Visual Basic.NET code. Paste **INSIDE** the ReportStart event.

```
Me.subReport1.Report = New rptOrders()
```

To write the code in C#

1. Click in the gray area below rptCustomers to select it.
2. Click the events icon in the Properties Window to display available events for the report.
3. Double-click **ReportStart**. This creates an event-handling method for the report's ReportStart event.
4. Add code to the handler to create a new instance of the subreport.

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

C# code. Paste **INSIDE** the ReportStart event.

```
this.subReport1.Report = new rptOrders();
```

Subreports with XML Data

Using XML data requires some setup that is different from other types of data. This walkthrough illustrates how to set up a subreport bound to the XML DataSource in the parent report.

This walkthrough is split up into the following activities:

- Connecting the parent report to an XML data source
- Adding controls to display the data
- Adding code to create a new instance of the subreport
- Adding code to pass a subset of the parent report's data to the subreport

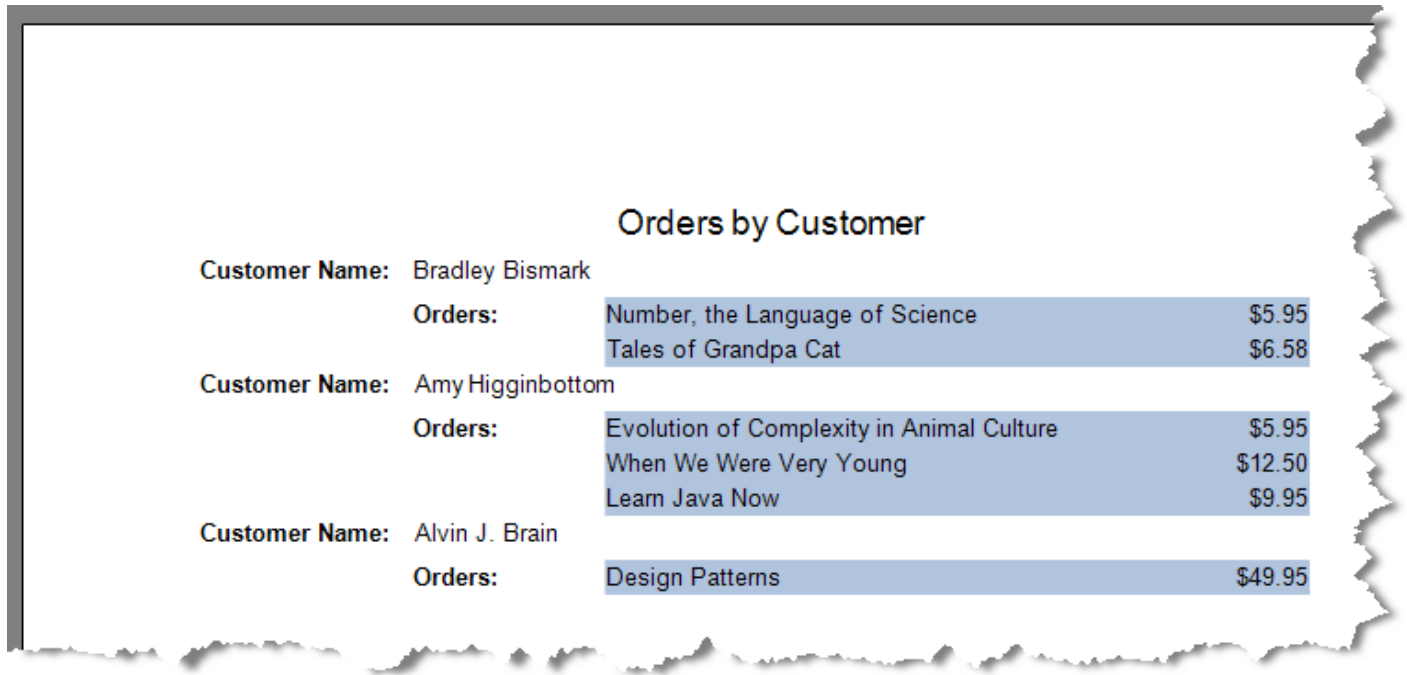


Tip: For basic steps like viewing a report, please see the **Basic Data Bound Reports** walkthrough.

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

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\customer.xml (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\customer.xml).

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



To connect the parent report to a data source

1. Add two ActiveReports to a Visual Studio project, naming them **rptMain** and **rptSub**.
2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

PageHeader1

Detail1

3. In the tabs along the top of the dialog, select **XML**.
4. Click the ellipsis button beside **File URL** to browse for the access path to **Customer.xml** and click **Open**. (The default installation path is C:\Program Files\GrapeCity\ActiveReports 6\Data\customer.xml, or C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\customer.xml on a **64-bit Windows operating system**).
5. In the **Recordset Pattern** field, enter **//CUSTOMER**.
6. Click **OK** to return to the report design surface.
7. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node, then **Document**, then **Customer** to see the fields.

To add controls to rptMain to display data

1. Set the **Height** property of the page header section to **0.3**.
2. Set the **CanShrink** property of the Detail section to **True** to eliminate white space.
3. Drag the following controls from the ActiveReports Toolbox onto the indicated section of rptMain, setting the properties as indicated.

Controls for rptMain

| Control | Section | Text | DataField | Miscellaneous | Location | Size |
|---------|------------|--------------------|-----------|--------------------------------------|----------|--------------|
| Label | PageHeader | Orders by Customer | | Font size = 14 Alignment = Center | 0, 0 in | 6.5, 0.25 in |

| | | | | | |
|-----------|--------|----------------|------|--------------|---------------|
| Label | Detail | Customer Name: | Bold | 0, 0 in | 1.2, 0.198 in |
| TextBox | Detail | NAME | | 1.2, 0 in | 2, 0.198 in |
| Label | Detail | Orders: | Bold | 1.2, 0.25 in | 1, 0.198 in |
| Subreport | Detail | | | 2.3, 0.25 in | 4, 1 in |

To add controls to rptSub to display data

1. Set the **CanShrink** property of the Detail section of rptSub to **True** to eliminate white space.
2. Set the **BackColor** property of the Detail section to **LightSteelBlue** to distinguish the subreport from the main report.



Tip: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

3. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
4. Add the following controls to the Detail section of rptSub, setting the properties as indicated:

Controls for rptSub

| Control | DataField | Name | Miscellaneous | Location | Size |
|---------|-----------|----------|---|----------|------------------|
| TextBox | TITLE | txtTitle | | 0, 0 in | 2.9, 0.198 in |
| TextBox | PRICE | txtPrice | Alignment = Right OutputFormat = \$#,##0.00 (or select Currency in the dialog) | 3, 0 in | 1, 0.198 in |

To add code to create a new instance of the subreport



Warning: Do not create a new instance of the subreport in the **Format** event. Doing so creates a new subreport each time the section Format code is run, which uses a lot of memory.

To write the code in Visual Basic

1. Right-click the design surface of **rptMain** and select **View Code**.
2. At the top left of the code view of the report, click the drop-down arrow and select **(rptMain Events)**.
3. At the top right of the code window, click the drop-down arrow and select **ReportStart**. This creates an event-handling method for the ReportStart event.
4. Add code to the handler to create an instance of rptSub.

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

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim rpt As rptSub
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
rpt = New rptSub
```

To write the code in C#

1. Click in the gray area below **rptMain** to select it.
2. Click the events icon in the Properties Window to display available events for the report.

3. Double-click **ReportStart**. This creates an event-handling method for the report's ReportStart event.
4. Add code to the handler to create a new instance of rptSub.

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

C# code. Paste JUST ABOVE the ReportStart event.

```
private rptSub rpt;
```

C# code. Paste INSIDE the ReportStart event.

```
rpt = new rptSub();
```

To add code to pass a subset of the parent report's data to the subreport

1. Double-click in the detail section of the design surface of rptMain to create a detail_Format event.
2. Add code to the handler to:
 - Create a new DataDynamics XMLDataSource
 - Type cast the new data source as rptMain's data source and set the NodeList to the "ORDER/ITEM" field
 - Display rptSub in the subreport control
 - Pass the new data source to the subreport

To write the code in Visual Basic

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

Visual Basic.NET code. Paste INSIDE the Format event.

```
Dim xmlDS As New DataDynamics.ActiveReports.DataSources.XMLDataSource
xmlDS.NodeList = CType(CType(Me.DataSource,
DataDynamics.ActiveReports.DataSources.XMLDataSource).Field("ORDER/ITEM", True),
System.Xml.XmlNodeList)
rpt.DataSource = xmlDS
Me.SubReport1.Report = rpt
```

To write the code in C#

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

C# code. Paste INSIDE the Format event.

```
DataDynamics.ActiveReports.DataSources.XMLDataSource xmlDS = new
DataDynamics.ActiveReports.DataSources.XMLDataSource();
xmlDS.NodeList = (System.Xml.XmlNodeList)
((DataDynamics.ActiveReports.DataSources.XMLDataSource)
this.DataSource).Field("ORDER/ITEM", true);
rpt.DataSource = xmlDS;
this.SubReport1.Report = rpt;
```


Hyperlinks for Simulated Drill-Down Reporting

Hyperlinks can be used in ActiveReports to pass values to parameters in other reports. 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:

- Connecting three reports to data sources

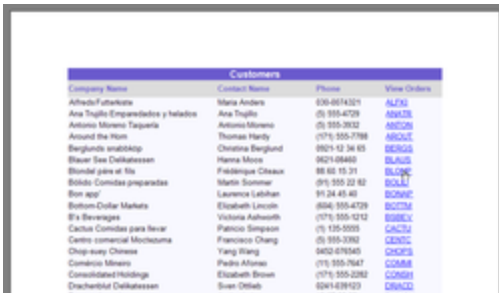
- Adding controls to each report to display data
- Creating a Windows Form Viewer
- Adding code to pass hyperlink values to parameters and open the drill-down report
- Adding code to set hyperlink properties to go back to the previous report

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

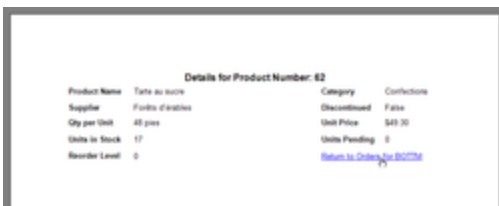
When you have finished this walkthrough, you will have reports that look similar to the following.



| Company Name | Contact Name | Phone | View Orders |
|-------------------------------------|--------------------|----------------|-----------------------|
| AffairsJaternata | Maria Anders | (505) 487-1511 | ALFKJ |
| Alex Trujillo Emparedados y helados | Alex Trujillo | (52) 555-4729 | ASACD |
| Antonio Moreno Tapacaria | Antonio Moreno | (5) 555-3932 | ANCON |
| Around the Horn | Thomas Hardy | (07) 555-7788 | ANSUL |
| Berglunds snabbkop | Christina Berglund | 0829 12 34 55 | BERGS |
| Beverly Hills Cookhouse | Hanna Moore | 9821 65440 | BLUSJ |
| Bonafide pates et sals | Felipeque Cleese | 88 66 15 31 | BLONC |
| Bottic Comidas preparadas | Marlin Sommer | (91) 555 22 82 | BOLLI |
| Bon app | Laurence Labarre | 91 34 45 45 | BONAP |
| Bottom Dollar Markets | Elsabeth Lincoln | (804) 555-4729 | BOTTL |
| B's Beverages | Victoria Ashworth | (07) 555-1212 | BSBEV |
| Cactus Comidas para llevar | Patricia Simpson | (7) 125-5555 | CACTU |
| Centro comercial Moctezuma | Francisco Chang | (5) 555-3389 | CENTC |
| Chop-eezy Chinese | Yang Wang | 8452 47545 | CHOPZ |
| Comércio Mineiro | Padre Almeida | (71) 555-7647 | COMBR |
| Consolidated Holdings | Elizabeth Brown | (71) 555-2182 | CONSH |
| Drachenklub Delikatessen | Sven Ottele | 0241 439123 | DRACK |



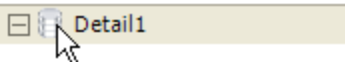
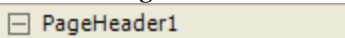
| Order Number | Order Date | Date Shipped | Product ID | Product Name | Unit Price | Quantity | Discount |
|--------------|------------|--------------|------------|----------------------|------------|----------|----------|
| 10388 | 01/20/95 | 01/24/95 | 82 | Buns | \$24.00 | 16 | 0% |
| | | | 83 | Outback Lager | \$12.00 | 30 | 0% |
| | | | 84 | Pate au saucis | \$18.25 | 15 | 0% |
| | | | 85 | Tarte au saucis | \$28.40 | 20 | 0% |
| 10415 | 02/15/95 | 02/15/95 | 22 | Getton | \$2.00 | 49 | 0% |
| | | | 23 | Raclette Courdaillon | \$44.00 | 16 | 0% |
| 10411 | 02/15/95 | 02/11/95 | 64 | Gula Malacca | \$15.50 | 40 | 20% |



| Product Name | Tarte au saucis | Category | Confections |
|----------------|----------------------|--|-------------|
| Supplier | F. Font's d'articles | Discontinued | False |
| Qty per Unit | 48 pieces | Unit Price | \$49.50 |
| Units in Stock | 17 | Units Pending | 0 |
| Reorder Level | 0 | Return to Orders for BOTTM | |

To connect three reports to data sources

1. Add three reports to a Visual Studio project, naming them **rptCustomers**, **rptOrders**, and **rptProductDetails**.
2. On **rptCustomers**, click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter the following SQL query

SQL Query


```
SELECT CompanyName, ContactName, Phone, CustomerID FROM Customers ORDER BY
CustomerID
```

8. Click **OK** to save the data source and return to the report design surface.
9. On **rptOrders**, click the gray report DataSource icon on the Detail section band and connect it to Nwind.mdb.
10. In the Query field, enter the following SQL query

SQL Query

```
SELECT Orders.OrderID, Orders.CustomerID, Orders.OrderDate, Orders.ShippedDate,
[Order Details].ProductID,
Products.ProductName, [Order Details].UnitPrice, [Order Details].Quantity, [Order
Details].Discount
FROM Products INNER JOIN (Orders INNER JOIN [Order Details] ON Orders.OrderID =
[Order Details].OrderID)
ON Products.ProductID = [Order Details].ProductID
WHERE Orders.CustomerID = '<%CustomerID||ALFKI%>'
ORDER BY Orders.OrderID, Products.ProductName
```

11. Click **OK** to save the data source and return to the report design surface.

 **Note:** The SQL queries for rptOrders and rptProductDetails use parameters syntax: '<%CustomerID||ALFKI%>' and '<%ProductID||1%>'. Using a default value allows the Report Explorer to populate so you can drag fields onto the report. For more information on parameters, see the **Parameters** topic.

12. On **rptProductDetails**, click the gray report DataSource icon on the Detail section band and connect it to Nwind.mdb.
13. In the Query field, enter the following SQL query

SQL Query

```
SELECT Products.ProductID, Products.ProductName, Suppliers.CompanyName,
Categories.CategoryName, Products.QuantityPerUnit,
Products.UnitPrice, Products.UnitsInStock, Products.UnitsOnOrder,
Products.ReorderLevel, Products.Discontinued
FROM Categories
INNER JOIN (Suppliers INNER JOIN Products ON Suppliers.SupplierID =
Products.SupplierID)
ON Categories.CategoryID = Products.CategoryID
WHERE Products.ProductID = <%ProductID||1%>
```

14. Click **OK** to save the data source and return to the report design surface.

To format the reports and add controls to display data

Top-level Report: rptCustomers

- In the design view of **rptCustomers**, click to select the page header section, and in the Properties Window, change the following properties:
 - **BackColor**: Gainsboro
 - **Height**: 0.5
- Select the detail section, and set its properties as follows:
 - **BackColor**: GhostWhite
 - **CanShrink**: True
- Add the following controls to the indicated sections rptCustomers, setting their properties as indicated:
rptCustomers controls

| Control | Section | Location | Size | DataField | Text | Miscellaneous |
|---------|------------|--------------|---------------|-------------|--------------|---|
| Label | PageHeader | 0, 0 in | 6.5, 0.198 in | | Customers | Alignment = Center BackColor = SlateBlue ForeColor = White Font Size = 12 Font Style = Bold |
| Label | PageHeader | 0, 0.26 in | 1.1, 0.198 in | | Company Name | ForeColor = SlateBlue Font Style = Bold |
| Label | PageHeader | 2.6, 0.26 in | 1, 0.198 in | | Contact Name | ForeColor = SlateBlue Font Style = Bold |
| Label | PageHeader | 4.3, 0.26 in | 1, 0.198 in | | Phone | ForeColor = SlateBlue Font Style = Bold |
| Label | PageHeader | 5.5, 0.26 in | 1, 0.198 in | | View Orders | ForeColor = SlateBlue Font Style = Bold |
| TextBox | Detail | 0, 0 in | 2.55, 0.2 in | CompanyName | | |
| TextBox | Detail | 2.6, 0 in | 1.6, 0.2 in | ContactName | | |
| TextBox | Detail | 4.3, 0 in | 1, 0.2 in | Phone | | |
| TextBox | Detail | 5.5, 0 in | 1, 0.2 in | CustomerID | | (Name) = txtCustomerID1 |

Mid-level Report: rptOrders

- In the design view of **rptOrders**, click to select the page header section, and set its **Height** property to **0.5**.
- Add the following controls to the page header section, setting their properties as indicated:
Page header controls for rptOrders

| Control | Location | Size | DataField | Text | Miscellaneous |
|---------|------------|---------------|--|---------------------|---|
| TextBox | 0, 0 in | 6.5, 0.198 in | = "Orders for Customer: " + CustomerID | | Alignment = Center Font Size = 12pt Font Style = Bold |
| Label | 0, 0.25 in | 6.5, 0.198 in | | Return to Customers | Alignment = Center HyperLink = Customers\\Main |

- Right-click the report and select **Insert**, then **GroupHeader/Footer**.
- Set the group header properties as follows:
 - **BackColor**: LightCyan

- **DataField:** OrderID
- **Height:** 0.5

5. Add the following controls to the group header section, setting their properties as indicated:

Group header controls for rptOrders

| Control | Location | Size | DataField | Text | Miscellaneous |
|---------|---------------|-------------|-------------|--------------|--|
| Label | 0, 0 in | 1, 0.198 in | | Order Number | BackColor = PaleTurquoise Font Style = Bold |
| TextBox | 1, 0 in | 0.5, 0.2 in | OrderID | | BackColor = PaleTurquoise |
| Label | 2.26, 0 in | 1, 0.198 in | | Order Date | Alignment = Right Font Style = Bold |
| TextBox | 3.34, 0 in | 1, 0.2 in | OrderDate | | Alignment = Right OutputFormat = MM/dd/yy |
| Label | 4.44, 0 in | 1, 0.198 in | | Date Shipped | Alignment = Right Font Style = Bold |
| TextBox | 5.5, 0 in | 1, 0.2 in | ShippedDate | | Alignment = Right OutputFormat = MM/dd/yy |
| Line | | | | | X1 = 0 X2 = 6.5 Y1 = 0.2 Y2 = 0.2 |
| Label | 0, 0.29 in | 1, 0.198 in | | Product ID | Font Style = Bold |
| Label | 1, 0.29 in | 1, 0.198 in | | Product Name | Alignment = Right Font Style = Bold |
| Label | 3.34, 0.29 in | 1, 0.198 in | | Unit Price | Alignment = Right Font Style = Bold |
| Label | 4.44, 0.29 in | 1, 0.198 in | | Quantity | Alignment = Right Font Style = Bold |
| Label | 5.5, 0.29 in | 1, 0.198 in | | Discount | Alignment = Right Font Style = Bold |

6. Click in the grey area below the report to select it, and set the **ShowParameterUI** property to **False** to avoid requesting parameters from the user.
7. Select the detail section, and set its **CanShrink** property to **True**.
8. From the Report Explorer, drag the following fields onto the detail section of rptOrders, setting their properties as indicated:

Detail section controls for rptOrders

| Field | Location | Size | Miscellaneous |
|-------------|------------|-------------|--|
| ProductID | 0, 0 in | 0.7, 0.2 in | Alignment = Right (Name) = txtProductID1 |
| ProductName | 1, 0 in | 2.2, 0.2 in | |
| UnitPrice | 3.34, 0 in | 1, 0.2 in | Alignment = Right OutputFormat = \$#,##0.00 |
| Quantity | 4.44, 0 in | 1, 0.2 in | Alignment = Right |
| Discount | 5.5, 0 in | 1, 0.2 in | Alignment = Right OutputFormat = 0% |

Lowest-Level Report: rptProductDetails

1. Add a textbox to the page header section of rptProductDetails, setting its properties as indicated:

rptProductDetails controls

| Control | Location | Size | DataField | Miscellaneous |
|---------|----------|-------------|---|---|
| TextBox | 0, 0 in | 6.5, 0.2 in | ="Details for Product Number: " + ProductID | Alignment = Center Font Style = Bold Font Size = 12pt |

2. Add the following labels to the detail section, setting their properties as indicated:

rptProductDetails labels

| Location | Size | Text | Miscellaneous |
|--------------|-------------|----------------|-------------------|
| 0, 0 in | 1, 0.198 in | Product Name | Font Style = Bold |
| 0, 0.28 in | 1, 0.198 in | Supplier | Font Style = Bold |
| 0, 0.54 in | 1, 0.198 in | Qty per Unit | Font Style = Bold |
| 0, 0.83 in | 1, 0.198 in | Units in Stock | Font Style = Bold |
| 0, 1.12 in | 1, 0.198 in | Reorder Level | Font Style = Bold |
| 4.4, 0 in | 1, 0.198 in | Category | Font Style = Bold |
| 4.4, 0.28 in | 1, 0.198 in | Discontinued | Font Style = Bold |
| 4.4, 0.54 in | 1, 0.198 in | Unit Price | Font Style = Bold |
| 4.4, 0.83 in | 1, 0.198 in | Units Pending | Font Style = Bold |

3. From the Report Explorer, drag the following fields onto the detail section, setting their properties as indicated:

rptProductDetails fields

| Field | Location | Size | Miscellaneous |
|------------------------------------|---------------|--------------|----------------------------|
| ProductName | 1.14, 0 in | 2.05, 0.2 in | |
| CompanyName | 1.14, 0.28 in | 2.05, 0.2 in | |
| QuantityPerUnit | 1.14, 0.54 in | 2.28, 0.2 in | |
| UnitsInStock | 1.14, 0.83 in | 1, 0.2 in | |
| ReorderLevel | 1.14, 1.12 in | 1, 0.2 in | |
| CategoryName | 5.5, 0 in | 1, 0.2 in | |
| Discontinued | 5.5, 0.28 in | 1, 0.2 in | |
| UnitPrice | 5.5, 0.54 in | 1, 0.2 in | OutputFormat = \$#,###0.00 |
| UnitsOnOrder | 5.5, 0.83 in | 1, 0.2 in | |
| TextBox | Location | Size | Miscellaneous |
| ="Return to Orders for " + OrderID | 4.4, 1.125 in | 2.1, 0.2 in | (Name) = txtReturnToOrders |

To add code to the viewer form for special hyperlink functions

This walkthrough assumes that you already know how to display a report in the viewer. For more information about these basic functions, see the **Viewing Reports** topic.

To write the code in Visual Basic

1. Right-click the form with the ActiveReports viewer and select **View Code**.
2. At the top left of the code window, click the drop-down arrow and select **Viewer1**.
3. At the top right of the code window, click the drop-down arrow and select **Hyperlink**. This creates an event-handling method for the form's Viewer1_Hyperlink event.
4. Add a method to clear the viewer and dispose of the report document.
5. Add code to the handler to process hyperlink text, determine which report to run, and display the report.

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

Visual Basic.NET code. Paste JUST ABOVE the Viewer HyperLink event.

```
Private LastCustID As String
```

Visual Basic.NET code. Paste JUST BELOW the Viewer HyperLink event.

```
Private Sub ClearViewer()
    Dim doc As DataDynamics.ActiveReports.Document.Document = viewer1.Document
    viewer1.Document = Nothing
    viewer1.Refresh()
    doc.Dispose()
    doc = Nothing
End Sub
```

Visual Basic.NET code. Paste INSIDE the Viewer HyperLink event.

```
'Process hyperlink text.
Dim report, parameter As String

If e.HyperLink.IndexOf("\") > 0 And e.HyperLink.Length > 2 Then
    report = e.HyperLink.Substring(0, e.HyperLink.IndexOf("\")).ToUpper()
    parameter = e.HyperLink.Substring(e.HyperLink.IndexOf("\") + 1)
Else
    MessageBox.Show("Cannot process hyperlink.")
    Return
End If

Dim rpt As DataDynamics.ActiveReports.ActiveReport = Nothing

'Determine which report to run.
If report.CompareTo("CUSTOMERS") = 0 Then
    rpt = New rptCustomers()
ElseIf report.CompareTo("ORDERS") = 0 Then
    rpt = New rptOrders()
    rpt.Parameters("CustomerID").Value = parameter
    LastCustID = parameter
ElseIf report.CompareTo("PRODUCTS") = 0 Then
    rpt = New rptProductDetails(LastCustID)
    rpt.Parameters("ProductID").Value = parameter
Else
    MessageBox.Show("Invalid report ID")
End If

'Check whether a report object exists. If so, run and display it.
If rpt IsNot Nothing Then
    ClearViewer()
    rpt.ShowParameterUI = False
```

```

    rpt.Run()
    Viewer1.Document = rpt.Document
    rpt.Dispose()
    rpt = Nothing
End If

```

To write the code in C#

1. Click the Viewer to select the it.
2. Click the events icon in the Properties Window to display available events for the viewer.
3. Double-click **Hyperlink**. This creates an event-handling method for the form's viewer1_Hyperlink event.
4. Add a method to clear the viewer and dispose of the report document.
5. Add code to the handler to process hyperlink text, determine which report to run, and display the report.

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

C# code. Paste **JUST ABOVE** the viewer HyperLink event.

```
private string LastCustID;
```

C# code. Paste **JUST BELOW** the viewer HyperLink event.

```
private void ClearViewer()
{
    DataDynamics.ActiveReports.Document.Document doc = viewer1.Document;

    viewer1.Document = null;
    viewer1.Refresh();

    doc.Dispose();
    doc = null;
}

```

C# code. Paste **INSIDE** the viewer HyperLink event.

```
//Process hyperlink text.
string report, parameter;

if (e.HyperLink.IndexOf('\') > 0 && e.HyperLink.Length > 2)
{
    report = e.HyperLink.Substring(0, e.HyperLink.IndexOf('\')).ToUpper();
    parameter = e.HyperLink.Substring(e.HyperLink.IndexOf('\') + 1);
}
else
{
    MessageBox.Show("Cannot process hyperlink.");
    return;
}

DataDynamics.ActiveReports.ActiveReport rpt = null;

//Determine which report to run.
if (report.CompareTo("CUSTOMERS") == 0)
{
    rpt = new rptCustomers();
}
else if (report.CompareTo("ORDERS") == 0)
{

```

```

    rpt = new rptOrders();
    rpt.Parameters["CustomerID"].Value = parameter;
    LastCustID = parameter;
}
else if (report.CompareTo("PRODUCTS") == 0)
{
    rpt = new rptProductDetails(LastCustID);
    rpt.Parameters["ProductID"].Value = parameter;
}
else
{
    MessageBox.Show("Invalid report ID");
}

//Check whether a report object exists. If so, run and display it.
if (rpt != null)
{
    ClearViewer();
    rpt.ShowParameterUI = false;
    rpt.Run();
    viewer1.Document = rpt.Document;
    rpt.Dispose();
    rpt = null;
}

```

To add code to set the drill-down hyperlink in rptCustomers

1. Double-click in the detail section to create a detail_Format event.
2. Add code to the event to set the hyperlink of the Customer ID textbox to call the viewer code that passes the parameter to rptOrders.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
txtCustomerID1.HyperLink = "Orders\" + txtCustomerID1.Text
```

To write the code in C#

C# code. Paste INSIDE the Format event.

```
txtCustomerID1.HyperLink = "Orders\\" + txtCustomerID1.Text;
```

To add code to set the drill-down hyperlink in rptOrders

1. Double-click in the detail section to create a detail_Format event.
2. Add code to the event to set the hyperlink of the Product ID textbox to call the viewer code that passes the parameter to rptProductDetails.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Format event.

```
txtProductID1.HyperLink = "Products\" + txtProductID1.Text
```

To write the code in C#

C# code. Paste INSIDE the Format event.

```
txtProductID1.HyperLink = "Products\\" + txtProductID1.Text;
```

To add code to create a parameter to hold the previous report for rptProductDetails

1. Right-click the report and select **View Code**.
2. Add code to create a parameter for the previous report.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **JUST BELOW** the Public Class rptProductDetails line.

```
'The prevRptParam member is used to track the parameter for  
'returning to the orders report for the correct customer  
Private prevRptParam As String  
Public Sub New(ByVal previousRptParameter As String)  
    InitializeComponent()  
    prevRptParam = previousRptParameter  
End Sub
```

To write the code in C#

C# code. Paste **JUST BELOW** the public partial class rptProductDetails() line.

```
//The prevRptParam member is used to track the parameter for  
//returning to the orders report for the correct customer  
private string prevRptParam;  
  
public rptProductDetails(string previousRptParameter)  
{  
    //  
    // Required for Windows Form Designer support  
    //  
    InitializeComponent();  
    prevRptParam = previousRptParameter;  
  
}
```

To add code to set the hyperlink to return to rptOrders in rptProductDetails

1. Double-click in the detail section to create a detail_Format event.
2. Add code to the event to set the hyperlink of the Return to Orders textbox to call the code that passes the parameter to rptOrders.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the Format event.

```
Me.txtReturnToOrders.Text = Me.txtReturnToOrders.Text + prevRptParam  
Me.txtReturnToOrders.HyperLink = "Orders\" + prevRptParam
```

To write the code in C#

C# code. Paste **INSIDE** the Format event.


```
this.txtReturnToOrders.Text = this.txtReturnToOrders.Text + prevRptParam;
this.txtReturnToOrders.HyperLink = "Orders\\" + prevRptParam;
```

Mail Merge with RichText

ActiveReports supports field merged reports using the RichText control. The RichText control can contain field place holders that can be replaced with 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:

- Connecting the report to a data source
- Adding controls and formatting the report
- Adding fields and text to the RichText control
- Using the FetchData event to conditionally format data
- Adding code to update RichText fields with current date and conditional values
- Adding code to send the group subtotal value to the RichText field

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.


A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).


When you complete this walkthrough, you will have a report that looks similar to the following:



To connect the report to a data source

1. Add a report to a Visual Studio project, naming it **rptLetter**.
2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

 PageHeader1

 Detail1

3. On the "OLE DB" tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the

appropriate access path.

6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter the following SQL query

SQL Query

```
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
```

8. Click **OK** to save the data source and return to the report design surface.

To add controls and format the report

1. Right-click the design surface of the report and select **Insert**, then **Group Header/Footer** to add group header and group footer sections.
2. In the Properties Window, make the following changes to the group header section:
 - **DataField:** CustomerID (This sets a new group for each customer.)
 - **Height:** 2.5
 - **KeepTogether:** True
3. Make the following changes to the group footer section:
 - **Height:** 1.1
 - **KeepTogether:** True
 - **NewPage:** After (This ensures that a new page begins after each customer's letter has finished rendering.)
4. Make the following changes to the detail section:
 - **CanShrink:** True
5. Make the following changes to the page header section:
 - **Height:** 0.8
6. Add the following controls to the PageHeader section and set the properties as indicated.

Page header controls

| Control | Miscellaneous | Size | Location |
|---------|--|--------------------|-----------|
| Label | Font Size = 36 Style = Bold Text = GrapeCity | 2.5, 0.65 in | 0, 0 in |
| Picture | PictureAlignment = TopLeft Image (click ellipsis, navigate to <i>C:\Program Files\GrapeCity\ActiveReports 6\Introduction (or to C:\Program Files (x86)\GrapeCity\ActiveReports 6\Introduction on a 64-bit Windows operating system)</i> and select itopimage1.gif) | 4.5, 0.65 in | 2.5, 0 in |


7. In the Report Explorer, expand the **Fields** node, then the **Bound** node. Drag the SubTotal field onto the group header section, add the following controls from the ActiveReports toolbox, and set the properties as indicated.

Group header controls

1, 0.2 in

| Control | DataField | Size | Text (Name) | Miscellaneous | Location |
|---------|-----------|-----------|------------------------|--|----------|
| Textbox | SubTotal | 1, 0.2 in | txtSubtotal1 (Name) | OutputFormat = Currency Visible = False SummaryType = SubTotal | 5, 0 in |

| | | | | |
|----------|-------------|------------|-----------------------------|----------------|
| RichText | 6.5, 2.1 in | | SummaryGroup = GroupHeader1 | |
| Label | 1, 0.198 in | Order ID | AutoReplaceFields = True | 0, 0 in |
| Label | 1, 0.198 in | Order Date | Bold | 0.875, 2.25 in |
| Label | 1, 0.198 in | Amount | Bold Alignment = Right | 1.875, 2.25 in |
| | | | | 4.375, 2.25 in |

 **Note:** Event though txtSubtotal1 is hidden, setting its properties is important as it provides the value and the formatting that is displayed in the RichText control.

8. In the Report Explorer, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the detail section and set the properties of each textbox as indicated.

Detail fields

| Field | Size | Miscellaneous | Location |
|-----------|-----------|--|-------------|
| OrderID | 1, 0.2 in | | 0.875, 0 in |
| OrderDate | 1, 0.2 in | OutputFormat = MM/dd/yy | 1.875, 0 in |
| Subtotal | 1, 0.2 in | OutputFormat = Currency Alignment = Right | 4.375, 0 in |

9. Add the following controls to the GroupFooter section and set the properties as indicated.

Group footer controls

| Control | Size | Text | Miscellaneous | Location |
|---------|----------------|---------------------|-------------------|---------------|
| Label | 1.35, 0.198 in | Best regards, | Alignment = Right | 5.15, 0.15 in |
| Label | 1.35, 0.198 in | Accounts Receivable | | 5.15, 0.8 in |

10. Add a label control to the PageFooter section and set the properties as indicated.

Page footer label

| Alignment | Size | Text | Location |
|-----------|---------------|---|----------|
| Center | 6.5, 0.198 in | GrapeCity, 401 Parkplace, Suite 411, Kirkland, WA 98033 | 0, 0 in |

To add fields to the RichText control

- Double-click inside the RichText control box and delete the default text.
- Right-click inside the box and choose **Insert Fields**.
- In the Insert Field dialog that appears, enter **Date** and click the **OK** button.
- Place the cursor in front of the text **[!Date]** that appears in the RichText control, and add spaces until the text is at the right edge of the control (but not overlapping to the next line).
- Place the cursor at the end of the text, and press the **Enter** key to move to the next line.
- Insert each of the following fields using the Insert Field dialog (see image below for arrangement of fields):
 - CompanyName
 - ContactName
 - Address
 - City
 - Region

- Country
- PostalCode
- ContactName

7. Add the following text to the RichText control box after all of the fields:

Paste into the RichText control

```
Dear [!ContactName],
```

```
Thank you for your business. Below is a list of your orders for the past year with a total of [!SubTotal].
```

```
Please take this opportunity to review each order and total for accuracy. Call us at 1-800-DNT-CALL with any questions or concerns.
```

8. Arrange the text and fields within the control as you would in any text editor to look like the following.

```
[CompanyName]
[Date]
[ContactName]
[Address]
[City][Region]
[Country] [PostalCode]

Dear [ContactName],

Thank you for your business. We have compiled a list of your orders for a total of [!SubTotal]. Please take this opportunity to review each order and amount for accuracy. Call us at (800) 555-1111 with any questions or concerns.
```

To use the FetchData event to conditionally format data

To write the code in Visual Basic

1. At the top left of the code view for the report, click the drop-down arrow and select **(rptLetter Events)**.
2. At the top right of the code window, click the drop-down arrow and select **FetchData**. This creates an event-handling method for the report's FetchData event.
3. Add code to the handler to add a comma and a space if there is a Region value for the customer's address.

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

Visual Basic.NET code. Paste JUST ABOVE the FetchData event.

```
Dim region As String
```

Visual Basic.NET code. Paste INSIDE the FetchData event.

```
'If there is no region for the customer, display nothing
If Fields("Region").Value Is System.DBNull.Value Then
    region = ""
Else
    'If there is a region, add a comma and a space
    region = ", " + Fields("Region").Value
End If
```

To write the code in C#

1. Back in design view, click in the gray area below the report to select it.
2. Click the events icon in the Properties window to display available events for the report.
3. Double-click **FetchData**. This creates an event-handling method for the report's FetchData event.
4. Add code to the handler to add a comma and a space if there is a Region value for the customer's address.

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

C# code. Paste JUST ABOVE the FetchData event.

```
string region;
```

C# code. Paste INSIDE the FetchData event.

```
if(Fields["Region"].Value is System.DBNull)
    region = "";
else
    region = ", " + Fields["Region"].Value.ToString();
```

To add code to update RichText fields with the current date and conditional values

1. Double-click in the group header section of the report to create an event-handling method for the group header's Format event.
2. Add code to the handler to:
 - Replace the Date field in the RichText control with the current system date
 - Replace the Region field with the conditional value created in the FetchData event

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Group Header Format event.**

```
'Use the current date in the letter
Me.RichTextBox1.ReplaceField("Date", System.DateTime.Today.Date.ToShortDateString())
'Use the value returned by the FetchData event
Me.RichTextBox1.ReplaceField("Region", region)
```

To write the code in C#**C# code. Paste INSIDE the Group Header Format event.**

```
//Use the current date in the letter
this.RichTextBox1.ReplaceField("Date", System.DateTime.Today.Date.ToShortDateString());
//Use the value returned by the FetchData event
this.RichTextBox1.ReplaceField("Region", region);
```

To add code to send the group subtotal value to the RichText field

1. Right-click in any section of the design window of rptLetter, and click on **View Code** to display the code view for the report.
2. At the top left of the code view for rptLetter, click the drop-down arrow and select **GroupHeader1**.
3. 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 GroupHeader1_BeforePrint event.



Note: We use the BeforePrint event instead of the Format event to get the final value of the subtotal field just prior to printing. For more information on section event usage, see the **Section Events** topic.

4. Add code to the handler to replace the value of the Subtotal field in the RichText control with the value of the hidden textbox in the group header.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Group Header BeforePrint event.**

```
'Use the value from the hidden group subtotal field  
Me.RichTextBox1.ReplaceField("SubTotal", Me.txtSubtotal1.Text)
```

To write the code in C#

C# code. Paste INSIDE the Group Header BeforePrint event.

```
//Use the value from the hidden group subtotal field  
this.RichTextBox1.ReplaceField("SubTotal", this.txtSubtotal1.Text);
```

Run Time or Ad Hoc Reporting

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.

Run Time Layouts

Describes how to create and modify report layouts dynamically.

Run Time Data Sources

Describes how to change the report data source at run time using the ReportStart event.

Run Time Layouts

ActiveReports objects and controls are completely accessible at run time. You can modify the properties of any of the report sections or controls to produce a dynamic report. The section Format event allows you to modify the properties of the section and its controls, including height, visibility, and other visual properties. The Format event is the only event in which you can modify the printable area of a section. Once this event has run, any changes to the section's height are not reflected in the report output. This walkthrough illustrates how to create a report layout at run time based on user input.



Note: Add controls dynamically in the **ReportStart** event. Otherwise, results may be unpredictable. For more information on events, see the **Sequence of Events** topic.

This walkthrough is split up into the following activities:

- Adding controls to the Windows Form to display fields and a viewer
- Generating a dataset for the Windows Form
- Adding code to create the report layout
- Adding code to fill the check list with fields and to launch the report
- Adding code to alternate colors in the detail section
- Adding code to the ReportStart event to call the report layout code
- Adding code to the button's Click event to collect the selected values and launch the report
- Adding code to enable the button when fields are selected
- Adding code to the Form_Load event to call the fill check list code

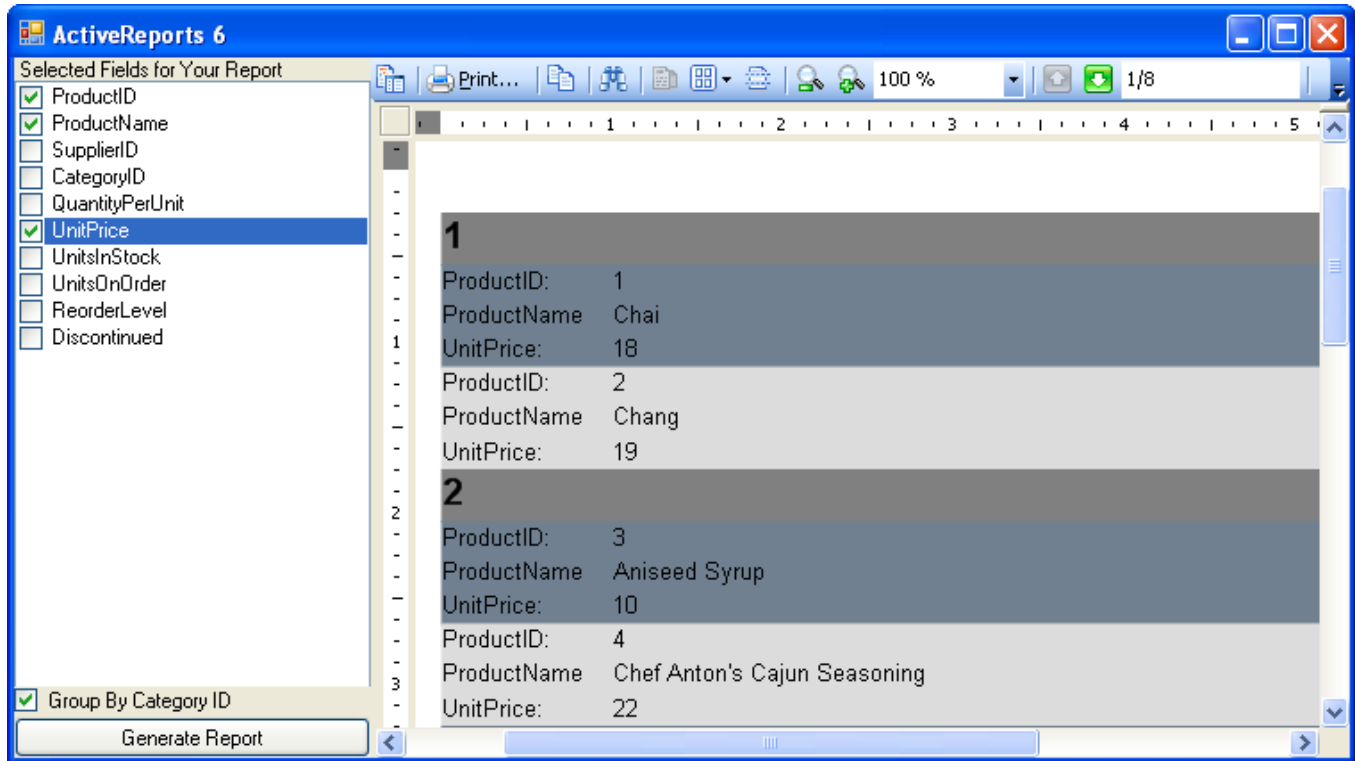


Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

When you have completed this walkthrough, you will have an application that looks similar to the following.



To add controls to the form

1. Resize the Windows form so that it is large enough to accommodate a number of controls.
2. From the Visual Studio toolbox, drag the following controls onto the form and set the properties as indicated.

Form controls

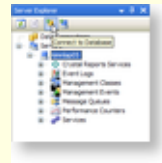
| Control | Location | Dock Property | Name | Text |
|-----------------|--------------|---------------|-----------------|-------------------------------|
| Panel | Windows Form | Left | Panel1 | |
| Label | Panel1 | Top | lblSelectFields | Select Fields for Your Report |
| Checked ListBox | Panel1 | Fill | clbFields | |
| Button | Panel1 | Bottom | btnGenRep | Generate Report |
| CheckBox | Panel1 | Bottom | chkGroup | Group By Category ID |
| Viewer | Windows Form | Fill | Viewer1 | |

To generate a dataset for the form

1. From the **Project** menu, select **Add New Item**.
2. Select **DataSet**, rename the file **NWINDDataSet.xsd** and click the **Add** button.
3. In the DataSet Designer that appears, click the **Server Explorer** link.
4. In the Server Explorer, expand the node for your local copy of the Northwind database, then the **Tables** node, and drag the **Products** table onto the DataSet designer.



Tip: If you do not see a copy of the Northwind database, click the **Connect to Database** icon and follow the prompts.



5. In the design view of your Windows form, expand the Data section of the Visual Studio Toolbox and double-click **DataSet** to open the **Add Dataset** dialog.
6. Under Typed dataset, select *YourProjectName.NWINDDataSet* and click **OK** to make the dataset available to your Windows form. **nwindDataSet1** appears in the tray below the form.

To add code to create the report layout

1. Right-click on **rptRunTime** and select **View Code**.
2. Add code within the class declaration of the report to:
 - Create an array of fields
 - Create an option for whether to use groups
 - Set properties on the report sections
 - Add textboxes and labels to the report based on the array of fields
 - Handle exceptions

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste **INSIDE** the class declaration of the report.

```
Private m_arrayFields As ArrayList
Private m_useGroups As Boolean
'Create an array to hold the fields selected by the user
Public WriteOnly Property FieldsList() As ArrayList
    Set(ByVal Value As ArrayList)
        m_arrayFields = Value
    End Set
End Property
'Create a property to hold the user's grouping choice
Public WriteOnly Property UseGroups() As Boolean
    Set(ByVal Value As Boolean)
        m_useGroups = False
        m_useGroups = Value
    End Set
End Property
Private m_defaultHeight As Single = 0.2F
Private m_defaultWidth As Single = 4.0F
Private m_currentY As Single = 0.0F
'Set up report formatting and add fields based on user choices
Private Sub constructReport()
    Try
        Me.Detail1.CanGrow = True
        Me.Detail1.CanShrink = True
        Me.Detail1.KeepTogether = True
        If m_useGroups = True Then
            'If the user wants grouping, add a group header and footer and set the grouping
            field
            Me.Sections.InsertGroupHF()
            CType(Me.Sections("GroupHeader1"), GroupHeader).DataField = "CategoryID"
            Me.Sections("GroupHeader1").BackColor = System.Drawing.Color.Gray
        End If
    Catch ex As Exception
        'Handle exceptions
    End Try
End Sub
```

```

        Me.Sections("GroupHeader1").CanGrow = True
        Me.Sections("GroupHeader1").CanShrink = True
        CType(Me.Sections("GroupHeader1"), GroupHeader).RepeatStyle =
RepeatStyle.OnPageIncludeNoDetail
        'Add a textbox to display the group's category ID
        Dim txt As New TextBox
        txt.DataField = "CategoryID"
        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
    Dim i As Integer
    For i = 0 To m_arrayFields.Count - 1
        'For all fields selected by the user (except CategoryID) create a label and a
textbox
        If m_arrayFields(i).ToString <> "CategoryID" Then
            Dim lbl As New Label
            'Set the label to display the name of the selected field
            lbl.Text = m_arrayFields(i) + ":"
            'Set the location of each label
            '(m_currentY gets the height of each control added on each iteration)
            lbl.Location() = New System.Drawing.PointF(0.0F, m_currentY)
            lbl.Width = 0.9F
            lbl.Height = m_defaultHeight
            Me.Detail1.Controls.Add(lbl)
            Dim txt As New TextBox
            'Set the textbox to display data
            txt.DataField = m_arrayFields(i)
            'Set the location of the textbox
            txt.Location = New System.Drawing.PointF(1.0F, m_currentY)
            txt.Width = m_defaultWidth
            txt.Height = m_defaultHeight
            Me.Detail1.Controls.Add(txt)
            'Set the textbox to use currency formatting if the field is UnitPrice
            If m_arrayFields(i) = "UnitPrice" Then
                txt.OutputFormat = "$#.00"
            End If
            'Increment the vertical location by adding the height of the added controls
            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

```

To write the code in C#

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

C# code. Paste INSIDE the class declaration of the report.

```

private ArrayList m_arrayFields;
//Create an array to hold the fields selected by the user
public ArrayList FieldsList
{
    set{m_arrayFields = value;}
}

```

```

}
private bool m_useGroups = false;
//Create a property to hold the user's grouping choice
public bool UseGroups
{
    set{m_useGroups = value;}
}
float m_defaultHeight = .2f;
float m_defaultWidth = 4f;
float m_currentY = 0f;
//Set up report formatting and add fields based on user choices
private void constructReport()
{
    try
    {
        this.detail.CanGrow = true;
        this.detail.CanShrink = true;
        this.detail.KeepTogether = true;
        if(m_useGroups)
        {
            //If the user wants grouping, add a group header and footer and set the grouping
field
            this.Sections.InsertGroupHF();
            ((GroupHeader)this.Sections["GroupHeader1"]).DataField = "CategoryID";
            this.Sections["GroupHeader1"].BackColor = System.Drawing.Color.Gray;
            this.Sections["GroupHeader1"].CanGrow = true;
            this.Sections["GroupHeader1"].CanShrink = true;
            ((GroupHeader)this.Sections["GroupHeader1"]).RepeatStyle =
RepeatStyle.OnPageIncludeNoDetail;
            this.Sections["GroupFooter1"].Height = 0;
            //Add a textbox to display the group's category ID
            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"))
            //For all fields selected by the user (except CategoryID) create a label and a
textbox
            {
                Label lbl = new Label();
                //Set the label to display the name of the selected field
                lbl.Text = m_arrayFields[i].ToString() + ":";
                //Set the location of each label
                //(m_currentY gets the height of each control added on each iteration)
                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();
                //Set the textbox to display data
                txt.DataField = m_arrayFields[i].ToString();
                //Set the location of the textbox
                txt.Location = new System.Drawing.PointF(1f,m_currentY);
                txt.Width = m_defaultWidth;
            }
        }
    }
}

```

```

        txt.Height = m_defaultHeight;
        this.detail.Controls.Add(txt);
        //Set the textbox to use currency formatting if the field is UnitPrice
        if (m_arrayFields[i].ToString().Equals("UnitPrice"))
        {
            txt.OutputFormat = "$#.00";
        }
        //Increment the vertical location by adding the height of the added controls
        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);
}
}
}

```

To add code to fill the check list with fields and to launch the report

1. Right-click the Windows Form and select **View Code**.
2. Add code within the class declaration of the form to:
 - Fill the check list with fields
 - Launch the report

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste **INSIDE** the class declaration of the form.

```

Dim i As Integer
Dim c As Integer
Dim m_arrayField As New ArrayList()
Private Sub fillCheckBox()
    For i = 0 To Me.NwindDataSet1.Tables.Count - 1
        For c = 0 To Me.NwindDataSet1.Tables(i).Columns.Count - 1
            Me.clbFields.Items.Add(Me.NwindDataSet1.Tables(i).Columns(c).ColumnName)
        Next
    Next
End Sub
Private Sub launchReport()
    Dim rpt As New rptRunTime()
    Dim dataAdapter As New NWINDDataSetTableAdapters.ProductsTableAdapter
    Try
        rpt.FieldsList = m_arrayField
        rpt.UseGroups = chkGroup.Checked
        dataAdapter.Fill(NwindDataSet1.Products)
        rpt.DataSource = Me.NwindDataSet1.Products
        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

```


To write the code in C#

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

C# code. Paste INSIDE the class declaration of the form.

```
ArrayList m_arrayField = new ArrayList();
private void fillCheckBox()
{
    for(int i = 0; i < this.nwindDataSet1.Tables.Count; i++)
    {
        for(int c = 0; c < this.nwindDataSet1.Tables[i].Columns.Count; c++)
        {
            this.clbFields.Items.Add(this.nwindDataSet1.Tables[i].Columns[c].ColumnName);
        }
    }
}
private void launchReport()
{
    try
    {
        rptRunTime rpt = new rptRunTime();
        rpt.FieldsList = m_arrayField;
        rpt.UseGroups = chkGroup.Checked;
        NWINDDataSetTableAdapters.ProductsTableAdapter dataAdapter = new
        NWINDDataSetTableAdapters.ProductsTableAdapter();
        dataAdapter.Fill(this.nwindDataSet1.Products);
        rpt.DataSource = this.nwindDataSet1.Products;
        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 alternate colors in the detail section

1. Double-click in the detail section of rptRunTime. This creates an event-handling method for rptRunTime's Detail_Format event.
2. Add code to the handler to alternate colors for a green bar report effect.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste JUST ABOVE the Detail Format event.

```
Dim m_count As Integer
```

Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
If m_count Mod 2 = 0 Then
    Me.Detail1.BackColor = System.Drawing.Color.SlateGray
Else
    Me.Detail1.BackColor = System.Drawing.Color.Gainsboro
End If
m_count = m_count + 1
```

To write the code in C#

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

C# code. Paste JUST ABOVE the Detail Format event.

```
int m_count;
```

C# code. Paste INSIDE the Detail Format event.

```
if(m_count % 2 == 0)
{
    this.detail.BackColor = System.Drawing.Color.SlateGray;
}
else
{
    this.detail.BackColor = System.Drawing.Color.Gainsboro;
}
m_count++;
```

Adding code to the ReportStart event to call the report layout code

1. Double-click in the gray area below rptRunTime to create an event-handling method for rptRunTime's ReportStart event.
2. Add code to call the constructReport method.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
constructReport()
```

To write the code in C#

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

C# code. Paste INSIDE the ReportStart event.

```
constructReport();
```

Adding code to the button's Click event to collect the selected values and launch the report

1. Double-click **btnGenRep** to create an event-handling method for its Click event.
2. Add code to the handler to collect the selected values and launch the report.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste INSIDE the button click event.

```
Me.m_arrayField.Clear()
For i = 0 To Me.clbFields.CheckedItems.Count - 1
    m_arrayField.Add(Me.clbFields.CheckedItems(i).ToString)
Next
launchReport()
```

To write the code in C#

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

C# code. Paste INSIDE the button click event.

```
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 enable the button when fields are selected

To write the code in Visual Basic.NET

1. At the top left of the code view for the form, click the drop-down arrow and select **clbFields**.
2. At the top right of the code window, click the drop-down arrow and select **SelectedIndexChanged**. This creates an event-handling method for the clbFields_SelectedIndexChanged event.
3. Add code to the handler to enable the button when fields are selected.

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

Visual Basic.NET code. Paste INSIDE the SelectedIndexChanged event.

```
If Me.clbFields.CheckedItems.Count < 0 Then
    Me.btnGenRep.Enabled = False
Else
    Me.btnGenRep.Enabled = True
End If
```

To write the code in C#

1. On the form, click **clbFields** to select it.
2. Click on the events icon in the Properties Window to display available events and double-click **SelectedIndexChanged**. This creates an event-handling method for the clbFields_SelectedIndexChanged event.
3. Add code to the handler to enable the button when fields are selected.

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

C# code. Paste INSIDE the SelectedIndexChanged event.

```
if(this.clbFields.CheckedItems.Count>0)
{
    this.btnGenRep.Enabled = true;
}
else
{
    this.btnGenRep.Enabled = false;
}
```

Adding code to the Form_Load event to call the fill check list code

1. Double-click the title bar of the form. This creates an event-handling method for the Windows Form_Load event.
2. Add code to the handler to call the fillCheckBox() method to populate clbFields with field values and to handle exceptions.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste INSIDE the Form Load event.

```
Try
```

```
fillCheckBox()  
Catch ex As Exception  
    System.Windows.Forms.MessageBox.Show(Me, "Error in Form1_Load: " + ex.Message, "Project  
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)  
End Try
```

To write the code in C#

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

C# code. Paste INSIDE the Form Load event.

```
try  
{  
    fillCheckBox();  
}  
catch(Exception ex)  
{  
    MessageBox.Show(this,"Error in Form1_Load: " + ex.Message,"Project Error",  
    MessageBoxButtons.OK,MessageBoxIcon.Error);  
}
```

Run Time Data Sources

ActiveReports allows you to change the data source of a report at run time. This walkthrough illustrates how to find the location of the sample database file on the user's computer and connect the report to it at run time.

This walkthrough is split up into the following activities:

- Connecting the report to a design time data source
- Adding controls to the report to display data
- Adding code to find the database path
- Adding code to change the data source at run time
- Adding code to close the data connection

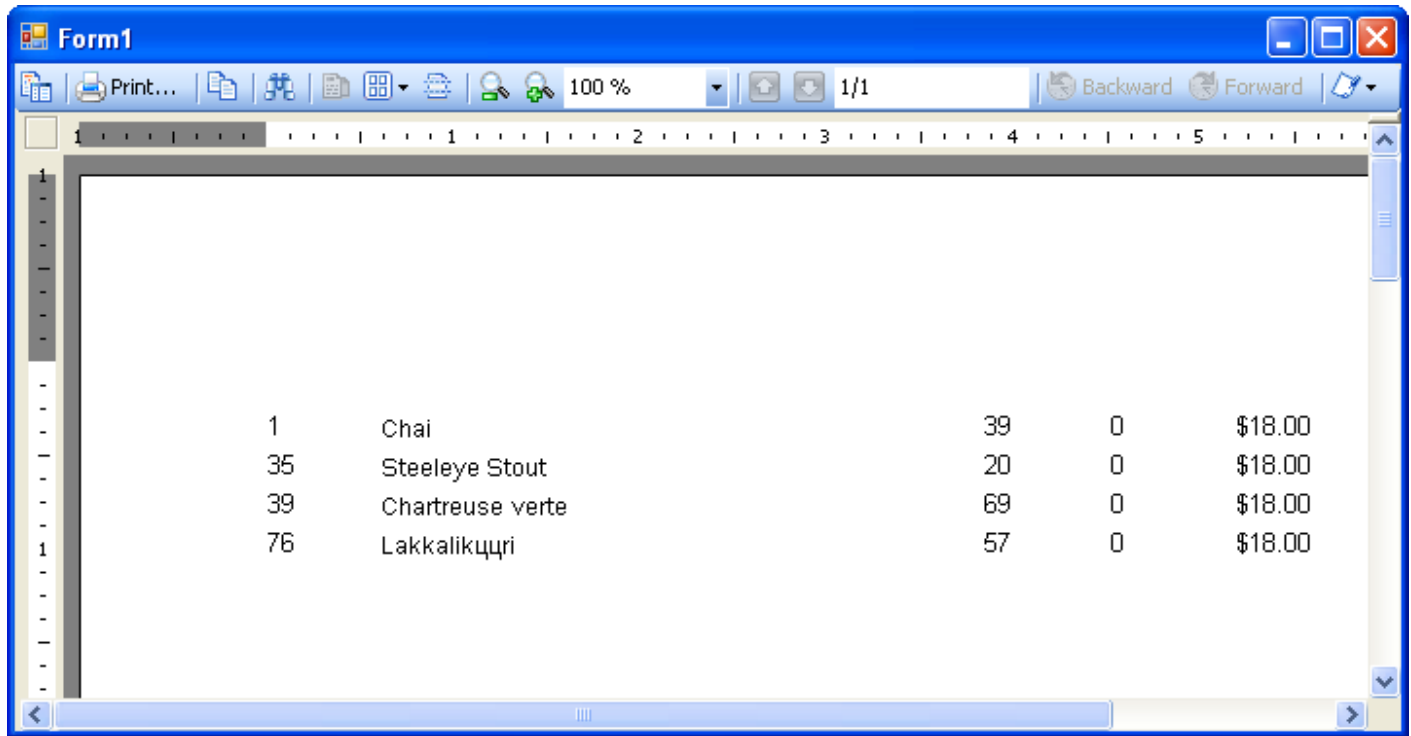


Tip: For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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



To connect the report to a design time data source

Tip: Even if you will change the data source at run time, setting a design time data source allows you to drag fields onto the report from the Report Explorer.

1. Add an **ActiveReport 6 (code-based) File** to a Visual Studio project, and rename the file **rptModifyDS**.
2. Click the gray report DataSource icon in the Detail section to open the report DataSource dialog.
3. On the **OLE DB** tab, click the **Build** button.
4. Select **Microsoft Jet 4.0 OLE DB Provider** and click **Next**.
5. Click the ellipsis button to browse for the access path to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to continue.
7. In the Query field, paste the following SQL query.

SQL Query

```
SELECT * FROM Products
```

8. Click **OK** to return to the report design surface.

To add controls to the report

1. Click the detail section to select it, and in the Properties Window, set the **CanShrink** property to **True**.
2. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node.
3. Drag the following fields onto the detail section of the report and set the properties of each textbox as indicated.

Detail section fields

| Field | Size | Location | Miscellaneous |
|-----------|-------------|----------|---------------|
| ProductID | 0.5, 0.2 in | 0, 0 in | |

| | | | |
|--------------|-------------|-----------|--|
| ProductName | 2.8, 0.2 in | 0.6, 0 in | |
| UnitsInStock | 0.5, 0.2 in | 3.5, 0 in | Alignment = Right |
| UnitsOnOrder | 0.5, 0.2 in | 4.1, 0 in | Alignment = Right |
| UnitPrice | 0.9, 0.2 in | 4.7, 0 in | OutputFormat = Currency Alignment = Right |

To find the database path

1. Right-click in any section of the design window of **rptModifyDS**, and select **View Code** to display the code view for the report.
2. Add code to the report to get the sample database path from the registry.

To write the code in Visual Basic

The following example shows what the code for the function looks like.

Visual Basic.NET code. Paste JUST BELOW the Imports DataDynamics.ActiveReports statements at the top of the code view.

```
Imports Microsoft.Win32
```

Visual Basic.NET code. Paste INSIDE the report class.

```
Private Function getDatabasePath() As String
    Dim regKey As RegistryKey
    regKey = Registry.LocalMachine
    regKey = regKey.CreateSubKey("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB")
    getDatabasePath = CType(regKey.GetValue(""), String)
End Function
```

To write the code in C#

C# code. Paste JUST BELOW the using DataDynamics.ActiveReports; statements at the top of the code view.

```
using Microsoft.Win32;
```

C# code. Paste INSIDE the report class.

```
private string getDatabasePath()
{
    RegistryKey regKey = Registry.LocalMachine;
    regKey = regKey.CreateSubKey("SOFTWARE\\GrapeCity\\ActiveReports 6\\SampleDB");
    return ((string) (regKey.GetValue("")));
}
```

To change the data source at run time

1. Double-click in the gray area below **rptModifyDS** to create an event-handling method for the **ReportStart** event.
2. Add code to the handler to change the data source at run time.

To write the code in Visual Basic.NET

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

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event.

```
Dim conn As System.Data.OleDb.OleDbConnection
Dim reader As System.Data.OleDb.OleDbDataReader
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Dim dbPath As String = getDatabasePath()
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb"
conn = New System.Data.OleDb.OleDbConnection(connString)
Dim cmd As New System.Data.OleDb.OleDbCommand("SELECT * FROM Products WHERE UnitPrice =
18", conn)
conn.Open()
reader = cmd.ExecuteReader()
Me.DataSource = reader
```

To write the code in C#

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

C# code. Paste JUST ABOVE the ReportStart event.

```
private static System.Data.OleDb.OleDbConnection conn;
private static System.Data.OleDb.OleDbDataReader reader;
```

C# code. Paste INSIDE the ReportStart event.

```
string dbPath = getDatabasePath();
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath +
"\NWIND.mdb";
conn = new System.Data.OleDb.OleDbConnection(connString);
System.Data.OleDb.OleDbCommand cmd = new System.Data.OleDb.OleDbCommand("SELECT * FROM
Products WHERE UnitPrice = 18", conn);
conn.Open();
reader = cmd.ExecuteReader();
this.DataSource = reader;
```

To close the data connection**To write the code in Visual Basic**

1. In design view of rptModifyDS, drop down the field at the top left of the code view and select (**rptModifyDS Events**).
2. Drop down the field at the top right of the code view and select **ReportEnd**. This creates an event-handling method for ReportEnd event.
3. Add code to the handler to close the data connection.

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

Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
reader.Close()
conn.Close()
```

To write the code in C#

1. Click in the gray area below rptModifyDS to select the report.
2. Click the events icon in the Properties Window to display available events for the report.

3. Double-click **ReportEnd**. This creates an event-handling method for the ReportEnd event.
4. Add code to the handler to close the data connection.

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

C# code. Paste INSIDE the ReportEnd event.

```
reader.Close();  
conn.Close();
```

Web Walkthroughs (Standard Edition)

Follow step-by-step tutorials as you create Visual Studio ASP.NET projects using ActiveReports. You can use the export controls to export and stream reports to a browser.

This section contains information about how to:

Custom Web Exporting (Std Edition) (on-line documentation)

Learn how to export reports to a memory stream and display the exports in a Web browser.

Custom HTML Outputter

Learn how to create a custom HTML outputter class and export reports to it.

Web Services

Learn how to create data or document Web services, and consume them in Windows applications.

Custom HTML Outputter

You can create a custom HTML outputter for your ActiveReports ASP.NET Web Application.

This walkthrough is split up into the following activities:

- Adding the Html Export to a Web project
- Creating a public class for the HTML outputter
- Adding code to export the report
- Adding a folder for report output

To add the Html Export control to the Web Form

1. From the Visual Studio **View** menu, select **Component Designer** to go to the design view of the ASPX file.
2. Drag the **HtmlExport** control from the Visual Studio toolbox onto the ASPX design view. See **Adding ActiveReports controls** for help if you need to add it to the toolbox.



Note: You can instead add a reference to **ActiveReports.HtmlExport** in the Solution Explorer if you prefer.

To create a public class for the HTML outputter

1. In the Solution Explorer window, right-click on your project name and select **Add**, then **New Item**.
2. In the **Add New Item** dialog that appears, select **Class**.
3. Change the name of the class to **MyCustomHtmlOutputter** and click the **Add** button.
4. This opens the code view of the class file where you can add the code needed to create the public class.
5. **For C# code**, add the **IOutputHtml** interface to **MyCustomHtmlOutputter** class.

C# code.


```
public class MyCustomHtmlOutputter: IOutputHtml
```

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste JUST ABOVE the class.

```
Imports System
Imports System.IO
Imports System.Web
Imports System.Text
Imports DataDynamics.ActiveReports
Imports DataDynamics.ActiveReports.Export.Html
```

Visual Basic.NET code. Paste INSIDE the class.

```
Implements IOutputHtml
'The http context of the request.
Private context As System.Web.HttpContext = Nothing
'The directory in which to save filename--this ensures that the filename
'is unique.
Private dirToSave As System.IO.DirectoryInfo = Nothing
Public mainPage As String = ""
Public Sub New(ByVal context As System.Web.HttpContext)
    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
            'Store 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
            'Create 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
        'Create 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
        '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)
    'Find 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 sourceStream into our buffer
    sourceStream.Read(buffer, 0, size)
    'Write out the 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

```

To write the code in C#

C# code. Paste JUST ABOVE the class.

```

using System;
using System.IO;
using System.Web;
using System.Text;
using DataDynamics.ActiveReports;
using DataDynamics.ActiveReports.Export.Html;

```

C# code. Paste INSIDE the class.

```

//The http context of the request
private System.Web.HttpContext context = null;
//The directory in which to save filename--this ensures that 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:
        {
            //Store 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:
    {
        // Create 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:
    {
        //Create a file with a .png extension:
        temp = this.GenUniqueFileNameWithExtension(".png");
        FileStream fs = File.Create(temp);
        this.WriteStreamToStream(info.OutputStream, fs);
        fs.Close();
        return temp;
    }

    default:
    {
        //Default to html:
        temp = this.GenUniqueFileNameWithExtension(".html");
        FileStream fs = File.Create(temp);
        this.WriteStreamToStream(info.OutputStream, fs);
        fs.Close();
        return temp;
    }
}

}

public void Finish()
{
}
#endregion

private void WriteStreamToStream(Stream sourceStream, Stream targetStream)
{
    //Find the size of the source stream
    int size = (int)sourceStream.Length;

    //Create a buffer that same size
    byte[] buffer = new byte[size];

    //Move the source stream to the beginning
    sourceStream.Seek(0, SeekOrigin.Begin);

    //Copy the sourceStream into our buffer
    sourceStream.Read(buffer, 0, size);
}
```

```

    //Write out the buffer to the target stream
    targetStream.Write(buffer, 0, size);
}

/// <summary>
/// Generates a unique file name with the specified extension.
/// </summary>
/// <param name="extensionWithDot">
/// The file extension begins with a dot such as ".jpg".
/// </param>
/// <returns></returns>
private string GenUniqueFileNameWithExtension(string extensionWithDot)
{
    System.Random r = new Random();
    bool unique = false;
    string filePath = "";
    int iRandom = 0;
    //Generate a random name until it's unique
    while(!unique)
    {
        iRandom = r.Next();
        //Buld the full filename
        System.Text.StringBuilder sb = new System.Text.StringBuilder();
        sb.Append(this.dirToSave.FullName);
        sb.Append(Path.DirectorySeparatorChar);
        sb.Append(iRandom.ToString());
        sb.Append(extensionWithDot);
        filePath = sb.ToString();
        unique = !File.Exists(filePath);
    }
    return filePath;
}

```

To add code to the Web Form to export to HTML

1. Add an ActiveReport to the project, and name it **rptCustHTML**.
2. Double-click on the design view of the ASPX. This creates an event-handling method for the Web Form's Page Load event.
3. Add the following code to the Page Load event.

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste **INSIDE** the Page Load event.

```

Dim rpt As New rptCustHTML()
Try
    rpt.Run(False)
Catch eRunReport As Exception
    'If the report fails to run, 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 the resulting data of this page should be
'cached since this could be a dynamic report that changes upon each request.
Response.Cache.SetCacheability(HttpCacheability.NoCache)

'Tell the browser this is an Html document so it will use an appropriate viewer.
Response.ContentType = "text/HTML"

'Create the Html export object
If HtmlExport1 Is Nothing Then
    HtmlExport1 = New DataDynamics.ActiveReports.Export.Html.HtmlExport()
End If
Dim outputter As New MyCustomHtmlOutputter(Me.Context)
HtmlExport1.Export(rpt.Document, outputter, "")
Response.Redirect("ReportOutput" + "/" +
System.IO.Path.GetFileName(outputter.mainPage))

```

To write the code in C#

C# code. Paste INSIDE the Page Load event.

```

rptCustHTML rpt = new rptCustHTML();
try
{
    rpt.Run(false);
}
catch (Exception eRunReport)
{
    //If the report fails to run, 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 the resulting data of this page should
//be cached since this could be a dynamic report that changes upon each request.
Response.Cache.SetCacheability(HttpCacheability.NoCache);

//Tell the browser this is an Html document so it will use an appropriate viewer.

```

```
Response.ContentType = "text/html";

//Create the HTML export object
if (this.htmlExport1 == null)
{
    this.htmlExport1 = new DataDynamics.ActiveReports.Export.Html.HtmlExport();
}

//Export the report to HTML in this session's webcache
MyCustomHtmlOutputter outputter = new MyCustomHtmlOutputter(this.Context);
this.htmlExport1.Export(rpt.Document, outputter, "");
Response.Redirect("ReportOutput" + "/" +
System.IO.Path.GetFileName(outputter.mainPage));
```

To add a folder to the project for report output

1. In the Solution Explorer, right-click your solution and select **Add**, then **New Folder**.
2. Name the folder **ReportOutput**.
3. Ensure that you have write permissions for this folder.
4. To view the results in your Web browser, run the project.

Web Services

ActiveReports provides support for web services to be used to return a dataset as a data source for a report or to return an ActiveReport document to show in a Windows Forms viewer. The following walkthroughs show how to create a simple web service for each scenario and how to create a Windows client application for each web service.

DataSet Web Service

Describes how to set up a simple web service that returns a dataset.

DataSet Windows Application


Describes how to set up a Windows client application for the dataset Web Service.

Document Web Service

Describes how to set up a simple web service that returns an ActiveReports document.

Document Windows Application

Describes how to set up a Windows client application for the ActiveReports Document Web Service.

 **Important:** In order to consume Web services in your Windows applications, you must set permissions to allow the ASP.NET user to consume the services. Ask your server administrator for help with this.

DataSet Web Service

With ASP.NET, you can set up a Web Service that returns a dataset to use for an ActiveReport. This walkthrough illustrates how to create one.

This walkthrough is split up into the following activities:

- Creating an ASP.NET Web Service project
- Adding code to create the Web method
- Testing the Web service
- Publishing the Web service
- Creating a virtual directory in IIS

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

To create an ASP.NET Web Service project

1. From the **File** menu, select **New**, then **Web Site**.
2. In the **New Web Site** dialog that appears, select **ASP.NET Web Service**.
3. Change the name of the project.
4. Click **OK** to open the new project in Visual Studio.

To create the Web Method

In the App_Code/Service.vb or Service.cs file that displays by default, replace the existing <WebMethod()> _ and HelloWorld function with code like the following.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste OVER the existing WebMethod.

```
Private connString As String
<WebMethod(Description:="Returns a DataSet containing all Products")> _
Public Function GetProduct() As Data.DataSet
    connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\nwind.mdb"
    Dim adapter As New Data.OleDb.OleDbDataAdapter("select * from products", connString)
    Dim ds As New Data.DataSet()
    adapter.Fill(ds, "Products")
    Return ds
End Function
```

To write the code in C#

C# code. Paste OVER the existing WebMethod.

```
private static string connString = "Provider=Microsoft.Jet.OLEDB.4.0;C:\Program
Files\GrapeCity\ActiveReports 6\Data\nwind.mdb";
[WebMethod(Description="Returns a DataSet containing all Products")]
public Data.DataSet GetProduct()
{
    Data.OleDb.OleDbDataAdapter adapter;
    Data.DataSet ds;
    adapter = new Data.OleDb.OleDbDataAdapter("select * from products", connString);
    ds = new Data.DataSet();
    adapter.Fill(ds, "Products");
    return ds;
}
```

To test the Web Service

1. Press **F5** to run the project.
2. If the Debugging Not Enabled dialog appears, select the option that enables debugging and click **OK** to continue.
3. In the list of supported operations, click the **GetProduct** link. (The description string from the code above appears below the link.)
4. Click the **Invoke** button to test the Web Service operation.
5. If the test is successful, a valid XML schema of the Northwind products table displays in a new browser window.

6. Copy the URL from the browser for use in the Web Reference of your **DataSet Windows Application**.

To publish the Web Service


1. In the Solution Explorer, right-click the project name and select **Publish Web Site**.
2. In the Publish Web Site window that appears, click the **OK** button.
3. This makes the Web Service available for consumption by other projects in a folder under the main folder called **PrecompiledWeb**.

To create a virtual directory in Internet Information Services

1. In Windows Explorer, navigate to the folder containing your Web service folder.
2. Right-click the folder and select **Sharing and Security**.
3. On the **Web Sharing** tab, select the **Share this folder** radio button.
4. Click the **OK** button to save the setting and close the window.

To create a virtual directory in IIS 7.x

1. In the Control Panel, open **Administrative Tools**, then **Internet Information Services Manager**.
2. In the **Internet Information Services Manager** window that appears, expand the tree view in the left pane until you see the Web Site you need to create a virtual directory for.
3. Right-click the node for the Web Site and select **Add Application...**
4. In the **Add Application** dialog box that appears, enter the following information:
 - Alias:** enter a name for the virtual directory.
 - Physical path:** select the folder that contains your Web service folder.
5. Click the **OK** button to save the settings and close the window.

 **Important:** In order to consume Web services in your Windows applications, you must set permissions to allow the ASP.NET user to consume the services. Ask your server administrator for help with this.

For information on consuming the DataSet Web Service in an ActiveReport, see **DataSet Windows Application**.

DataSet Windows Application

You can use a Web Service that returns a dataset as the data source for your reports in Windows applications. This walkthrough illustrates how to create a Windows client application that uses the dataset Web Service as the data source for an ActiveReport.

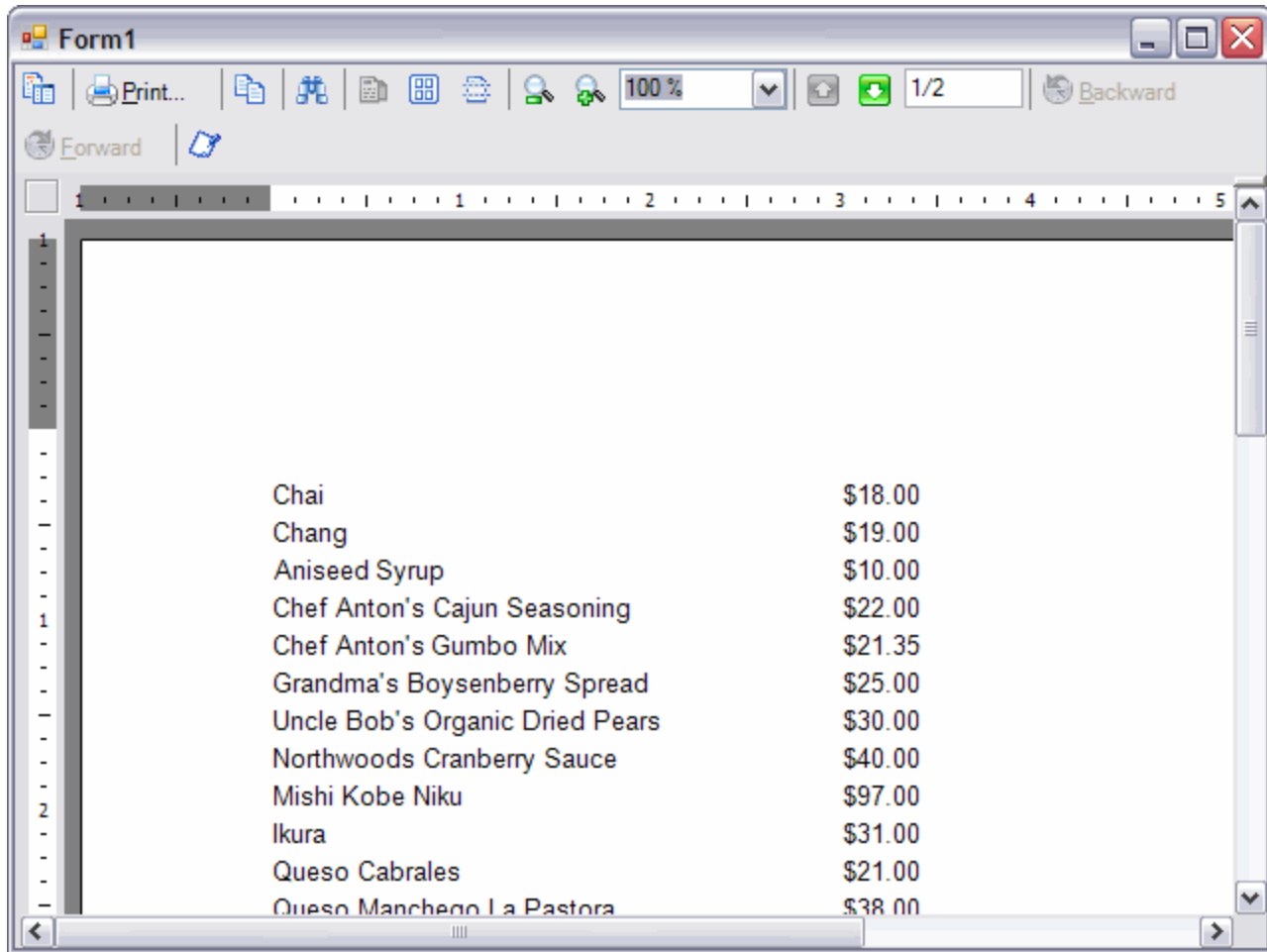
This walkthrough builds on the **DataSet Web Service** walkthrough and is split up into the following activities:

- Adding controls to a report
- Adding a reference to a Web service to the project
- Setting the report data source to the one returned by the Web service

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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



To add controls to a report

1. Add a new ActiveReport to a Windows application.
2. Add the following controls to the Detail section of the report, setting their properties as indicated:
Detail section controls

| Control | DataField | Miscellaneous | Location |
|---------|-------------|-------------------------|----------|
| Textbox | ProductName | Size = 2.9, 0.19 | 0, 0 |
| Textbox | UnitPrice | OutputFormat = Currency | 3, 0 |

3. Set the **CanShrink** property of the Detail section to **True**.

To add a reference to a web service to the project

To add a reference to a web service in Visual Studio 2005

1. From the **Project** menu, select **Add Web Reference**.
2. In the **Add Web Reference** window that appears, click the **Web services on the local machine** link.
3. Click the link to the virtual directory you created in the previous walkthrough. You can get the address by running the project from the previous walkthrough and copying the url from the address in the browser. (It will look something like `http://localhost:####/DataSetWS/Service.asmx` where `####` is the port number.)
4. Click the **Go** button, and then click the **Add Reference** button when the Web Service is recognized.

To add a reference to a web service in Visual Studio 2008 or 2010 that is compatible with .NET Framework 2.0 Web service

1. From the **Project** menu, select **Add Service Reference**.
2. In the **Add Service Reference** window that appears, click **Advanced** button.
3. In the **Service Reference Settings** window that appears, click **Add Web Reference** button.
4. In the **Add Web Reference** window that appears, click the **Web services on the local machine** link.
5. Click the link to the virtual directory you created in the previous walkthrough. You can get the address by running the project from the previous walkthrough and copying the url from the address in the browser. (It will look something like `http://localhost:####/DataSetWS/Service.asmx` where `####` is the port number.)
6. Click the **Go** button, and then click the **Add Reference** button when the Web Service is recognized.

To add a reference to a web service in Visual Studio 2008 or 2010

1. From the **Project** menu, select **Add Service Reference**.
2. In the **Add Service Reference** window that appears, type in the address of the virtual directory you created in the previous walkthrough. You can get the address by running the project from the previous walkthrough and copying the url from the address in the browser. (It will look something like `http://localhost:####/DataSetWS/Service.asmx` where `####` is the port number.)
3. Click the **Go** button, and then click the **OK** button when the Web Service is recognized.

To set the report data source to the one returned by the Web service**To set the report data source (for Visual Studio 2005 and Visual Studio 2008 or 2010 compatible with .NET Framework 2.0 Web service)**

1. Double-click the gray area below the report. This creates an event-handling method for the ReportStart event.
2. Add code to the handler to use the web service dataset in the report.
The following example shows what the code for the method looks like.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the ReportStart event**

```
Dim ws As New localhost.Service
Dim ds As DataSet() = ws.GetProduct()
Me.DataSource = ds
Me.DataMember = "Products"
```

To write the code in C#**C# code. Paste INSIDE the ReportStart event.**

```
localhost.DataSetWS ws = new localhost.Service;
DataSet ds = ws.GetProduct();
this.DataSource = ds;
this.DataMember = "Products";
```

To set the report data source (for Visual Studio 2008 or 2010)

1. Double-click the gray area below the report. This creates an event-handling method for the ReportStart event.
2. Add code to the handler to use the web service dataset in the report.
The following example shows what the code for the method looks like.

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the ReportStart event.**

```
Dim ws As New ServiceReference1.ServiceSoapClient()
```


```
Dim ds As DataSet = ws.GetProduct()  
Me.DataSource = ds  
Me.DataMember = "Products"
```

To write the code in C#

C# code. Paste INSIDE the ReportStart event.

```
ServiceReference1.ServiceSoapClient ws = new  
ServiceReference1.ServiceSoapClient();  
DataSet ds = ws.GetProduct();  
this.DataSource = ds;  
this.DataMember = "Products";
```

To update the app.config file (Visual Studio 2008 or 2010 project)

 **Note:** You need to update the app.config file if you added the Service Reference to the Visual Studio 2008 or 2010 project in the previous section.

1. In the Solution Explorer, open the app.config file.
2. In the tag <binding name = "ServiceSoap"...>, set **maxBufferSize** and **maxReceivedMessageSize** to some large number, for example, 200500.
3. In the next tag <readerQuotas...>, set **maxArrayLength** to some large number, for example, 60500.

Document Web Service

With ASP.NET and ActiveReports, you can set up a Web Service that returns a report document which can be shown in a report viewer control.

This walkthrough illustrates how to create a Web Service that returns the contents of an ActiveReport as a byte array.

This walkthrough is split up into the following activities:

- Creating an ASP.NET Web Service project
- Adding a report and connecting it to data
- Adding controls to the report
- Adding code to create the Web Method
- Testing the Web Service
- Publishing the Web Service
- Creating a virtual directory in IIS

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit operating Windows system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

You must also add a reference in your project to the **System.Data.OleDb** namespace.

When you have completed this walkthrough, you will have a Web Service that returns the contents of an ActiveReport as a byte array.

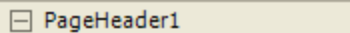
To create an ASP.NET Web Service project

1. From the Visual Studio **File** menu, select **New**, then **Web Site**.
2. In the Templates window of the **New Web Site** dialog, select **ASP.NET Web Service**.

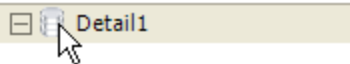
3. Change the name of the project from **WebSite1** to **ARDocumentWS**.
4. Click **OK** to open the new project in Visual Studio.

To add a report and connect it to data

1. From the **Website** menu, select **Add New Item**.
2. Select **ActiveReports 6 (code-based) File**, rename it **rptProducts**, and click the **Add** button.
3. In the message box that appears, click the **Yes** button to place the report inside the App_Code folder. Instead of the design view, the code view of the report opens.
4. To go to the design view of the report, in the Solution Explorer, right-click **rptProducts** and select **View Designer**.
5. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1



Detail

6. On the **OLE DB** tab, next to Connection String, click the **Build** button.
7. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
8. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
9. Click **OK** to close the window and fill in the Connection String field.
10. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Products INNER JOIN Categories ON Products.CategoryID =
Categories.CategoryID ORDER BY CategoryName
```

11. Click **OK** to save the data source and return to the report design surface.

To add controls to the report

1. Right-click on the design surface of the report and select **Insert**, then **Group Header/Footer** to add a group header and group footer section.
2. Click to select the group header section, and in the Properties window, make the following changes:
 - Change the **Name** property to **ghCategories**
 - Change the **DataField** property to **CategoryName**
 - Change the **BackColor** property to **LightGray**
3. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node.
4. Drag the following field onto ghCategories and set the properties as indicated.
 Field for ghCategories

| Field | Size | Location | Miscellaneous |
|--------------|-------------|----------|---|
| CategoryName | 6.5, 0.2 in | 0, 0 in | Font size = 14 ForeColor = DarkGreen |

5. Add a second GroupHeader/Footer section to the report to contain labels.
6. Make the following changes to the new group header:
 - Change the **Name** property to **ghProducts**
 - Change the **BackColor** property to **WhiteSmoke**

7. Add labels with the following properties to ghProducts:

Labels for ghProducts

| Name | Text | Location |
|------------------------|----------------|-----------|
| lblProductName | Product Name | 0, 0 in |
| lblUnitPrice | Unit Price | 2.4, 0 in |
| lblUnitsInStock | Units in Stock | 4, 0 in |
| lblUnitsOnOrder | Units on Order | 5.5, 0 in |

8. Set the **CanShrink** property of the detail section to **True**.
9. From the Report Explorer, drag the following fields onto the detail section and set the properties as indicated.

Controls for the detail section

| Field | Text | Miscellaneous | Location |
|--------------|----------------|--|-----------|
| ProductName | Product Name | Size = 2.25, 0.2 in | 0, 0 in |
| UnitPrice | Unit Price | OutputFormat = Currency Alignment = Right | 2.4, 0 in |
| UnitsInStock | Units In Stock | Alignment = Right | 4, 0 in |
| UnitsOnOrder | Units On Order | Alignment = Right | 5.5, 0 |

To write the code to create the Web Method

1. On the **Service.vb** or **Service.cs** tab is the code view of the Service.asmx file.
2. Replace the existing WebMethod and HelloWorld function with the following code.

To write the code in Visual Basic.NET

Visual Basic.NET code. REPLACE the existing WebMethod and function with this code.

```
<WebMethod( _
    Description:="Returns a products report grouped by category")> _
Public Function GetProductsReport() As Byte()
    Dim rpt As New rptProducts()
    rpt.Run()
    Return rpt.Document.Content
End Function
```

To write the code in C#

C# code. REPLACE the existing WebMethod and function with this code.

```
[WebMethod(Description="Returns a products report grouped by category")]
public Byte[] GetProductsReport()
{
    rptProducts rpt = new rptProducts();
    rpt.Run();
    return rpt.Document.Content;
}
```

To test the Web Service

1. Press **F5** to run the project. The Service page appears in your browser.
2. In the list of supported operations at the top, click **GetProductsReport**.
3. Click the **Invoke** button to test the Web Service operation.
4. If the test is successful, you will see the binary version of the contents of rptProducts.

To publish the Web Service

1. In the Solution Explorer, right-click the project name and select **Publish Web Site**.
2. Click the **OK** button. The Web Service is now available for consumption by other projects.

To create a virtual directory in Internet Information Services

1. In Windows Explorer, navigate to the folder containing your Web service folder.
2. Right-click the folder and select **Sharing and Security**.
3. On the **Web Sharing** tab, select the **Share this folder** radio button.
4. Click the **OK** button to save the setting and close the window.

To create a virtual directory in IIS 7.x

1. In the Control Panel, open **Administrative Tools**, then **Internet Information Services Manager**.
2. In the **Internet Information Services Manager** window that appears, expand the tree view in the left pane until you see the Web Site you need to create a virtual directory for.
3. Right-click the node for the Web Site and select **Add Application...**
4. In the **Add Application** dialog box that appears, enter the following information:
 - Alias:** enter a name for the virtual directory.
 - Physical path:** select the folder that contains your Web service folder.
5. Click the **OK** button to save the settings and close the window.



Important: In order to consume Web services in your Windows applications, you must set permissions to allow the ASP.NET user to consume the services. Ask your server administrator for help with this.

For information on consuming the Document Web Service in a viewer, see **Document Windows Application**.

Document Windows Application

In ActiveReports 6, you can use a Web Service that returns the content of an ActiveReport to show in the Windows Forms viewer control.

This walkthrough illustrates how to create a Windows client application that returns the content of an ActiveReport in the Windows Forms viewer.

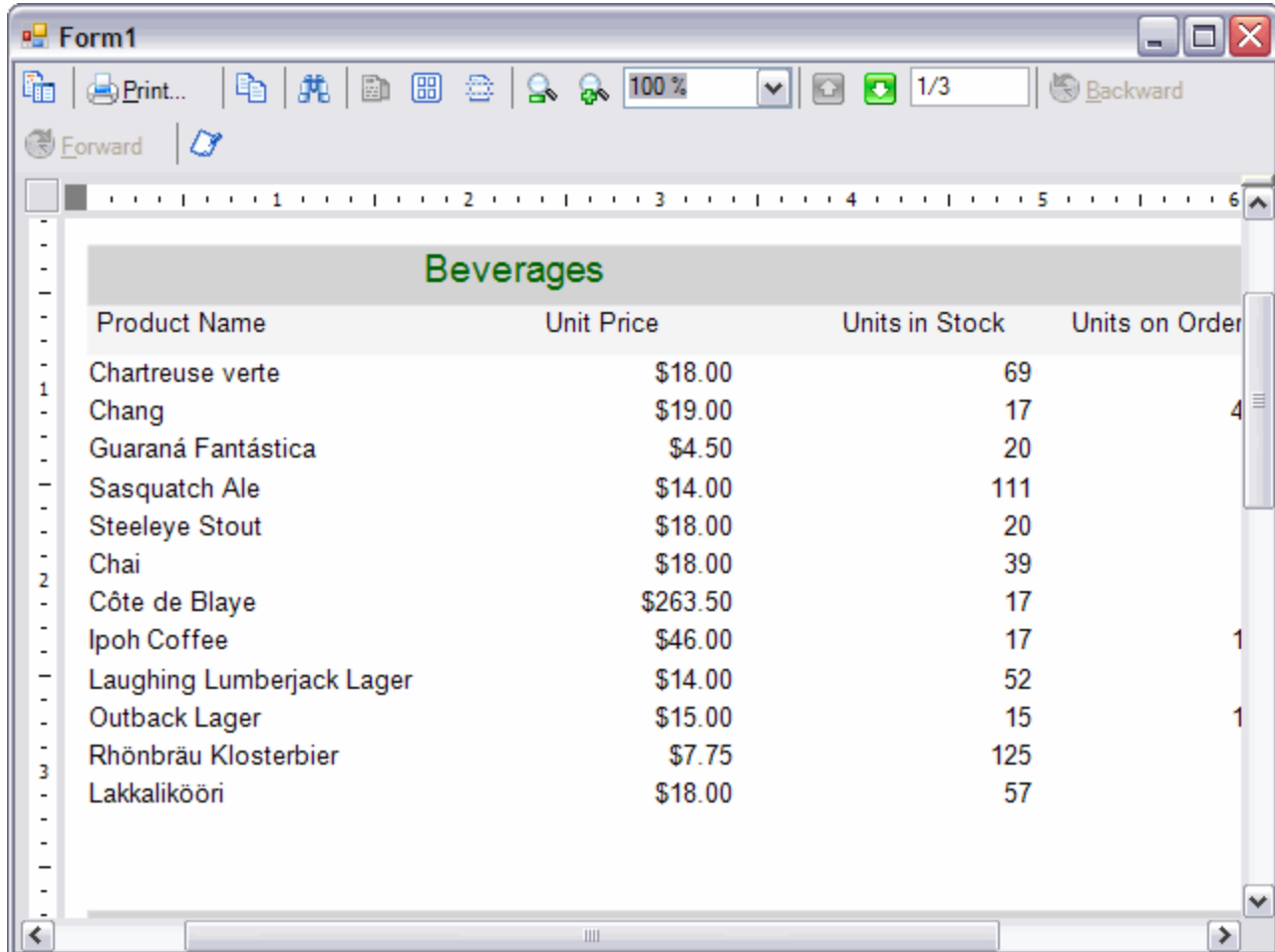
This walkthrough builds on the **Document Web Service** walkthrough and is split up into the following activities:

- Creating a Visual Studio project
- Adding the ActiveReports Windows Forms viewer control to the form
- Adding a reference to a Web service to the project
- Displaying the content returned by the Document Web Service in the viewer
- Running the project

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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



| Beverages | | | |
|---------------------------|------------|----------------|----------------|
| Product Name | Unit Price | Units in Stock | Units on Order |
| Chartreuse verte | \$18.00 | 69 | |
| Chang | \$19.00 | 17 | 4 |
| Guaraná Fantástica | \$4.50 | 20 | |
| Sasquatch Ale | \$14.00 | 111 | |
| Steeleye Stout | \$18.00 | 20 | |
| Chai | \$18.00 | 39 | |
| Côte de Blaye | \$263.50 | 17 | |
| Ipoh Coffee | \$46.00 | 17 | 1 |
| Laughing Lumberjack Lager | \$14.00 | 52 | |
| Outback Lager | \$15.00 | 15 | 1 |
| Rhönbräu Klosterbier | \$7.75 | 125 | |
| Lakkalikööri | \$18.00 | 57 | |

To create a Visual Studio project

1. From the **File** menu, select **New**, then **Project**.
2. In the Templates section of the New Project dialog, select **Windows Application**.
3. Change the name of the application to **ARDocumentClient**.
4. Click **OK** to open the project.

To add the ActiveReports viewer control

1. From the Visual Studio toolbox, drag the ActiveReports **Viewer** control onto the form.
2. Change the **Dock** property for the viewer control to **Fill**, and resize the form to accommodate a report.

To add a web reference

To add a reference to a web service in Visual Studio 2005

1. From the **Project** menu, select **Add Web Reference**.
2. Type in the address of the .asmx file for the ActiveReports Document Web Service you created in the previous walkthrough. For example: <http://localhost/ARDocumentWS/Service.asmx>
3. Click the **Add Reference** button when the Web Service is recognized.

To add a reference to a web service in Visual Studio 2008 or 2010 that is compatible with .NET Framework 2.0 Web service

1. From the **Project** menu, select **Add Service Reference**.
2. In the **Add Service Reference** window that appears, click the **Advanced** button.
3. In the **Service Reference Settings** window that appears, click **Add Web Reference** button.
4. From the **Project** menu, select **Add Web Reference**.
5. Type in the address of the .asmx file for the ActiveReports Document Web Service you created in the previous walkthrough. For example: <http://localhost/ARDocumentWS/Service.asmx>
6. Click the **Add Reference** button when the Web Service is recognized.

To add a reference to a web service in Visual Studio 2008 or Visual Studio 2010

1. From the **Project** menu, select **Add Service Reference**.
2. In the **Add Service Reference** that appears, type in the address of the .asmx file for the ActiveReports Document Web Service you created in the previous walkthrough. For example: <http://localhost/ARDocumentWS/Service.asmx>
3. Click the **Go** button, and then click the **OK** button when the Web Service is recognized.

To display the content returned by the Document Web Service in the viewer**To display the content (for Visual Studio 2005 and Visual Studio 2008 or 2010 compatible with .NET Framework 2.0 Web service)**

1. Double-click Form1 to create an event-handling method for the Form1_Load event.
2. Add code to the handler to display the document Web service content in the viewer.

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

To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Form Load event.**

```
Dim ws As New localhost.Service
Me.Viewer1.Document.Content = ws.GetProductsReport()
```

To write the code in C#**C# code. Paste INSIDE the Form Load event.**

```
localhost.Service ws = new localhost.Service();
this.viewer1.Document.Content = ws.GetProductsReport();
```

To display the content (for Visual Studio 2008 or 2010)

1. Double-click on Form1 to create an event-handling method for the Form1_Load event.
2. Add code to the handler to display the document Web service content in the viewer.

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


To write the code in Visual Basic.NET**Visual Basic.NET code. Paste INSIDE the Form Load event.**

```
Dim ws As New ServiceReference1.ServiceSoapClient()
Me.Viewer1.Document.Content = ws.GetProductsReport()
```

To write the code in C#**C# code. Paste INSIDE the Form Load event.**

```
ServiceReference1.ServiceSoapClient ws = new ServiceReference1.ServiceSoapClient();  
this.viewer1.Document.Content = ws.GetProductsReport();
```

To update the app.config file (Visual Studio 2008 or 2010 project)

 **Note:** You need to update the app.config file if you added the Service Reference to the Visual Studio 2008 or 2010 project in the previous section.

1. In the Solution Explorer, open the app.config file.
2. In the tag <binding name = "ServiceSoap"...>, set **maxBufferSize** and **maxReceivedMessageSize** to some large number, for example, 200500.
3. In the next tag <readerQuotas...>, set **maxArrayLength** to some large number, for example, 60500.

To run the project

- Press **F5** to run the project.

Layout Files with Embedded Script

ActiveReports allows you to embed script in reports so that code becomes portable when you save a report layout to XML-based RPX format. This characteristic allows the options of stand-alone reporting and web reporting without the need to distribute related .vb or .cs files. By embedding script when the report is saved as an RPX file, it can later be loaded, run and displayed directly to the viewer control without using the designer. Script can also be used in conjunction with RPX files to allow distributed reports to be updated without recompiling the Visual Studio project.

Script for Simple Reports

Describes how to embed script in a simple stand-alone report.

Script for Subreports


Describes how to embed script to pass a parameter to a subreport.

Script for Simple Reports

ActiveReports allows you to use scripting to embed code in reports saved to the XML-based RPX file format. By embedding script in reports saved as RPX files, you can later load, run, and display reports directly in the viewer control without using the designer. This walkthrough illustrates how to include scripting in a simple report.

This walkthrough is split into the following activities:

- Temporarily connecting the report to a data source
- Adding controls to a report to display data
- Adding scripting to supply data for the controls
- Saving the report to an RPX file

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.


A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\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 Products: | |
| | 12 |
| 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 |



To temporarily connect the report to a data source

1. Add an **ActiveReports 6 File (xml-based)** to a Visual Studio project and rename it **rptScript**.

 **Note:** The following steps are just for convenience so that the fields list in the Report Explorer can be populated at design time.

2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.

 PageHeader1

  Detail1

3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter the following SQL query.

SQL Query

```
SELECT * FROM categories INNER JOIN products ON categories.categoryid =
```

```
products.categoryid ORDER BY products.categoryid, products.productid
```

- Click **OK** to save the data source and return to the report design surface.

To add controls to the report

- Right-click on the design surface of the report and select **Insert** then **Group Header/Footer** to add group header and footer sections to your report.
- Increase the group header section's height so that you have room to work.
- In the Properties Window, make the following changes to the group header:
 - BackColor:** LightBlue
 - CanShrink:** True
 - DataField:** CategoryID
 - GroupKeepTogether:** All
 - KeepTogether:** True
- Add the following controls to the GroupHeader section and set the properties as indicated.

Group header controls

| Control | DataField | Text | Location | Size | Miscellaneous |
|---------|--------------|----------------|-------------|-------------|--|
| TextBox | CategoryName | | 0, 0 in | 6.5, 0.2 in | BackColor = CadetBlue Font Style = Bold Font Size = 12 |
| TextBox | Description | | 0, 0.2 in | 6.5, 0.2 in | BackColor = CadetBlue |
| Label | | Product Name | 0, 0.4 in | 1, 0.2 in | Font Style = Bold |
| Label | | Units in Stock | 5.5, 0.4 in | 1, 0.2 in | Font Style = Bold Alignment = Right |

- In the Report Explorer, expand the Fields node, then the Bound node, and drag the following fields onto the detail section, setting the properties as indicated.

Detail section fields

| Control | DataField | Location | Size | Alignment |
|---------|--------------|-----------|---------------|-----------|
| TextBox | ProductName | 0, 0 in | 5.5, 0.198 in | |
| TextBox | UnitsInStock | 5.5, 0 in | 1, 0.198 in | Right |

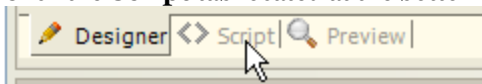
- Set the **CanShrink** property of the detail section to **True**.
- Select both of the text boxes in the detail section, right-click and select **Format Border**.
 - Select dark cyan in the color combo box
 - Select the solid line in the Line Styles pane
 - Click the bottom edge in the Preview pane
 - Click the **OK** button to add a solid cyan line to the bottom edge of the text boxes.
- Increase the group footer section's height so that you have room to work.
- Make the following changes to the group footer:
 - BackColor:** PaleGreen
 - CanShrink:** True
- Add the following controls to the GroupFooter section, setting the properties as indicated.

Group footer controls

| Control | DataField | Size | Miscellaneous | Location |
|---------|-------------|---------------|---|----------|
| TextBox | TotalLabel | 3, 0.198 in | Font Style = Bold | 2.5, 0 |
| TextBox | ProductName | | SummaryType = Subtotal SummaryFunc = Count SummaryRunning = Group SummaryGroup = GroupHeader1 Alignment = Right | 5.5, 0 |
| Label | | 6.5, 0.198 in | BackColor = White (creates white space after the subtotal) Delete default text from Text property | 0, 0.25 |

To add scripting to the report to supply data for the controls

1. Click in the grey area below the report to select it, and in the Properties Window, change the **ScriptLanguage** property for the report to the scripting language you want to use. The default setting is **C#**.
2. Click the **Script** tab located at the bottom edge of the report designer to access the scripting editor.



3. Add the scripting code.

The following example shows what the scripting code looks like.

Warning: Do not access the Fields collection outside the **DataInitialize** and **FetchData** events. Accessing the Fields collection outside of these events is not supported, and has unpredictable results.

To write the script in Visual Basic.NET.

Visual Basic.NET script. Paste in the script editor window.

```
Private Shared m_reader As System.Data.OleDb.OleDbDataReader
Private Shared m_cnn As System.Data.OleDb.OleDbConnection

Public Sub ActiveReport_ReportStart()
    'Set up a data connection for the report
    rpt.DataSource = ""
    Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.mdb"
    Dim sqlString As String = "SELECT * FROM categories INNER JOIN products ON
categories.categoryid = products.categoryid ORDER BY products.categoryid,
products.productid"

    m_cnn = new System.Data.OleDb.OleDbConnection(connString)
    Dim m_Cmd As System.Data.OleDb.OleDbCommand = new
System.Data.OleDb.OleDbCommand(sqlString, m_cnn)

    If m_cnn.State = System.Data.ConnectionState.Closed Then
        m_cnn.Open
    End If
    m_reader = m_Cmd.ExecuteReader
End Sub

Public Sub ActiveReport_DataInitialize()
```

```

'Add data fields to the report
rpt.Fields.Add("CategoryID")
rpt.Fields.Add("CategoryName")
rpt.Fields.Add("ProductName")
rpt.Fields.Add("UnitsInStock")
rpt.Fields.Add("Description")
rpt.Fields.Add("TotalLabel")
End Sub

Public Function ActiveReport_FetchData(ByVal eof As Boolean) As Boolean
Try
    m_reader.Read
    'Populated the fields with data from the data reader
    rpt.Fields("CategoryID").Value = m_reader("categories.CategoryID")
    rpt.Fields("CategoryName").Value = m_reader("CategoryName")
    rpt.Fields("ProductName").Value = m_reader("ProductName")
    rpt.Fields("UnitsInStock").Value = m_reader("UnitsInStock")
    rpt.Fields("Description").Value = m_reader("Description")
    'Concatenate static text with data
    rpt.Fields("TotalLabel").Value = "Total Number of " + m_reader("CategoryName")+ "
Products:"
    eof = False
Catch
    'If the end of the data file has been reached, tell the FetchData function
    eof = True
End Try
Return eof
End Function

Public Sub ActiveReport_ReportEnd()
'Close the data reader and connection
m_reader.Close
m_cnn.Close
End Sub

```

To write the script in C#.

C# script. Paste in the script editor window.

```

//C#
private static System.Data.OleDb.OleDbDataReader m_reader;
private static System.Data.OleDb.OleDbConnection m_cnn;

public void ActiveReport_ReportStart()
{
    //Set up a data connection for the report
    rpt.DataSource = "";
    string m_cnnString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\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()
{
    //Add data fields to the report
    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();
        //Populated the fields with data from the data reader
        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();
        //Concatenate static text with data
        rpt.Fields["TotalLabel"].Value = "Total Number of " +
m_reader["CategoryName"].ToString() + " Products:";
        eof = false;
    }
    catch
    {
        //If the end of the data file has been reached, tell the FetchData function
        eof = true;
    }
    return eof;
}

public void ActiveReport_ReportEnd()
{
    //Close the data reader and connection
    m_reader.Close();
    m_cnn.Close();
}

```

To save the report to an XML-based RPX file

1. From the **Report** menu, select **Save Layout**.
2. In the Save dialog that appears, enter a name for the report, i.e. **rptScript.rpx**, and click the **Save** button.

Script for Subreports


ActiveReports allows you to use scripting to permit reports saved to an XML file to contain code. By including scripting

when reports 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:

- Temporarily connecting the main report to a data source
- Connecting the subreport to a data source
- Adding controls to each report to display data
- Adding the scripting code for rptMain
- Loading an xml-based report into the viewer

 **Tip:** For basic steps like adding a report to a Visual Studio project and viewing a report, please see the **Basic Data Bound Reports** walkthrough.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

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


| Alfreds Futterkiste | | | |
|---------------------|----------|------------|----------|
| Ordered: | 09/25/95 | Required: | 10/23/95 |
| | | Shipped: | 10/03/95 |
| Product Name | Quantity | Unit Price | Discount |
| Rössle Sauerkraut | 15 | \$45.60 | 25% |
| Chartreuse verte | 21 | \$18.00 | 25% |
| Spegesild | 2 | \$12.00 | 25% |

| Ordered: | 11/03/95 | Required: | 12/01/95 |
|--------------|----------|------------|----------|
| | | Shipped: | 11/13/95 |
| Product Name | Quantity | Unit Price | Discount |
| Vegie-spread | 20 | \$43.90 | 0% |

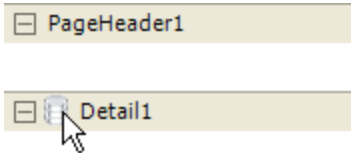
| Ordered: | 11/13/95 | Required: | 12/25/95 |
|---------------|----------|------------|----------|
| | | Shipped: | 11/21/95 |
| Product Name | Quantity | Unit Price | Discount |
| Aniseed Syrup | 6 | \$10.00 | 0% |
| Lakkalikööri | 15 | \$18.00 | 0% |

To temporarily connect the main report to a data source

1. Add an **ActiveReports 6 File (xml-based)** to a Visual Studio project and rename it **rptMain**.

 **Note:** The following steps are just for convenience so that the fields list in the Report Explorer can be populated at design time.

2. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



3. On the **OLE DB** tab, next to Connection String, click the **Build** button.
4. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
5. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
6. Click **OK** to close the window and fill in the Connection String field.
7. In the Query field, enter the following SQL query.


SQL Query

```
SELECT * FROM Orders INNER JOIN Customers ON Orders.CustomerID =
Customers.CustomerID ORDER BY CompanyName, OrderDate
```

8. Click **OK** to save the data source and return to the report design surface.

To temporarily connect the subreport to a data source

1. Add a second **ActiveReports 6 (xml-based) File** to the project and rename it **rptSub**.

 **Note:** The following steps are just for convenience so that the fields list in the Report Explorer can be populated at design time.

2. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
3. Click in the grey area below the report to select it, and in the Properties Window, change the report's **ShowParameterUI** property to **False**. This prevents the subreport from requesting a parameter from the user.
4. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.
5. Click the **Build** button.
6. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
7. Click the ellipsis button to browse to the NorthWind database. Click **Open** once you have selected the appropriate access path.
8. Click **OK** to close the window and fill in the Connection String field.
9. In the Query field, enter the following SQL query.

SQL Query

```
Select * from [order details] inner join products on [order details].productid =
products.productid
```

10. Click **OK** to return to the report design surface.

To add controls to rptMain to display data

1. Right-click on the design surface of rptMain and select **Insert** then **Group Header/Footer** to add group header and footer sections to the report.
2. In the Properties Window, make the following changes to the group header:
 - **Name:** ghCompanies
 - **BackColor:** LemonChiffon

- **CanShrink:** True
 - **DataField:** CompanyName
 - **GroupKeepTogether:** All
 - **KeepTogether:** True
3. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node. Drag the following field onto ghCompanies and set the properties as indicated.
- Group header ghCompanies field**

| Field | Miscellaneous | Size | Location |
|-------------|-------------------------------------|-----------|----------|
| CompanyName | Font style = Bold Font Size = 12 | 4, 0.2 in | 0, 0 in |

4. Add a second GroupHeader/Footer section to rptMain.
5. Make the following changes to the group header:
- **Name:** ghOrders
 - **BackColor:** LightYellow
 - **CanShrink:** True
 - **DataField:** OrderDate
 - **GroupKeepTogether:** All
 - **KeepTogether:** True
6. Drag the following fields and controls onto ghOrders and set the properties as indicated.
- ghOrders controls**

| Control | DataField | Size | Text | Miscellaneous | Location |
|---------|--------------|----------------|-----------|--|------------|
| TextBox | OrderDate | 1, 0.198 in | | OutputFormat = MM/dd/yy | 1.13, 0 in |
| TextBox | RequiredDate | 1, 0.198 in | | OutputFormat = MM/dd/yy | 3.5, 0 in |
| TextBox | ShippedDate | 1, 0.198 in | | OutputFormat = MM/dd/yy Alignment = Right | 5.5, 0 in |
| Label | | 1, 0.198 in | Ordered: | Font style = Bold | 0, 0 in |
| Label | | 1, 0.198 in | Required: | Font style = Bold | 2.5, 0 in |
| Label | | 0.65, 0.198 in | Shipped: | Font style = Bold | 4.8, 0 in |

7. Change the **CanShrink** property of the detail section to **True**.
8. Drag the following control onto the detail section and set the properties as indicated.
- Detail section control**

| Control | ReportName | Name | Size | Location |
|-----------|---------------------------------|------------|-----------|----------|
| Subreport | C:\full project path\rptSub.rpx | SubReport1 | 6.5, 1 in | 0, 0 in |

To add controls to rptSub

1. Right-click on the design surface of rptSub and select **Insert** then **Group Header/Footer** to add group header and footer sections to the report.
2. Make the following changes to the group header:
 - **Name:** ghOrderDetails
 - **BackColor:** LightSteelBlue
 - **CanShrink:** True
 - **DataField:** OrderID

3. Add four label controls to ghOrderDetails and set the properties as indicated.
ghOrderDetails labels

| Font style | Text | Alignment | Location |
|------------|--------------|-----------|----------|
| Bold | Product Name | Left | 0, 0 |
| Bold | Quantity | Right | 3.25, 0 |
| Bold | Unit Price | Right | 4.4, 0 |
| Bold | Discount | Right | 5.5, 0 |

4. Add four line controls to ghOrderDetails and set the properties as indicated.
ghOrderDetails line controls

| Name | X1 | X2 | Y1 | Y2 |
|-------|------|------|-----|-----|
| Line1 | 3.2 | 3.2 | 0 | 0.2 |
| Line2 | 4.3 | 4.3 | 0 | 0.2 |
| Line3 | 5.45 | 5.45 | 0 | 0.2 |
| Line4 | 0 | 6.5 | 0.2 | 0.2 |

5. Make the following changes to the detail section:

- **BackColor:** Gainsboro
- **CanShrink:** True

6. In the **Report Explorer**, expand the **Fields** node, then the **Bound** node. Drag the following fields onto the detail section and set the properties as indicated.

Detail section fields

| Field | Size | Alignment | OutputFormat | Location |
|----------------------------|----------------|-----------|--------------|------------|
| ProductName | 3.15, 0.198 in | Left | | 0, 0 in |
| Quantity | 1, 0.198 in | Right | | 3.25, 0 in |
| Products. UnitPrice | 1, 0.198 in | Right | Currency | 4.4, 0 in |
| Discount | 1, 0.198 in | Right | 0% | 5.5, 0 in |

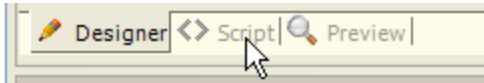
7. Add four line controls to the detail section and set the properties as follows (or copy and paste them from ghOrderDetails):

Detail section lines

| Name | X1 | X2 | Y1 | Y2 |
|-------|------|------|-----|-----|
| Line5 | 3.2 | 3.2 | 0 | 0.2 |
| Line6 | 4.3 | 4.3 | 0 | 0.2 |
| Line7 | 5.45 | 5.45 | 0 | 0.2 |
| Line8 | 0 | 6.5 | 0.2 | 0.2 |

To embed script in the main report

1. Change the **ScriptLanguage** property for the report to the appropriate scripting language. The default setting is C#.
2. Click the Script tab located below the report designer to access the scripting editor.



3. Embed script to set the data source for the main report and pass data into the subreport.

The following example shows what the script looks like.

To write the script in Visual Basic.NET

Visual Basic.NET script. Paste in the script editor window.

```
Dim rptSub As New DataDynamics.ActiveReports.ActiveReport()
Sub ActiveReport_ReportStart
    'Create a new instance of the generic report
    rptSub = new DataDynamics.ActiveReports.ActiveReport()
    'Load the rpx file into the generic report
    Dim xtr As New System.Xml.XmlTextReader(me.SubReport1.ReportName)
    rptSub.LoadLayout(xtr)
    xtr.Close()
    'Connect data to the main report
    Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB;Persist Security Info=False"
    Dim sqlString As String = "Select * from orders inner join customers on
orders.customerid = customers.customerid order by CompanyName,OrderDate"
    Dim ds As new DataDynamics.ActiveReports.DataSources.OleDBDataSource()
    ds.ConnectionString = connString
    ds.SQL = sqlString
    rpt.DataSource = ds
End Sub

Sub Detail1_Format
    Dim rptSubCtl As DataDynamics.ActiveReports.SubReport = me.SubReport1
    Dim childDataSource As New DataDynamics.ActiveReports.DataSources.OleDBDataSource()
    childDataSource.ConnectionString = CType(rpt.DataSource,
DataDynamics.ActiveReports.DataSources.OleDBDataSource).ConnectionString
    'Set a parameter in the SQL query
    childDataSource.SQL = "Select * from [order details] inner join products on [order
details].productid = products.productid where [order details].orderid = <%OrderID%>"
    'Pass the data to the subreport
    rptSub.DataSource = childDataSource
    'Display rptSub in the subreport control
    rptSubCtl.Report = rptSub
End Sub
```

To write the script in C#

C# code. Paste in the script editor window.

```
DataDynamics.ActiveReports.ActiveReport rptSub;
public void Detail1_Format()
{
    DataDynamics.ActiveReports.SubReport rptSubCtl = this.SubReport1;
    DataDynamics.ActiveReports.DataSources.OleDBDataSource childDataSource = new
DataDynamics.ActiveReports.DataSources.OleDBDataSource();
    childDataSource.ConnectionString =
((DataDynamics.ActiveReports.DataSources.OleDBDataSource)
rpt.DataSource).ConnectionString;
    //Set a parameter in the SQL query
```

```

    childDataSource.SQL = "Select * from [order details] inner join products on [order
details].productid = products.productid where [order details].orderid = <%OrderID%>";
    //Pass the data to the subreport
    rptSub.DataSource = childDataSource;
    //Display rptSub in the subreport control
    rptSubCtl.Report = rptSub;
}

public void ActiveReport_ReportStart()
{
    //Create a new instance of the generic report
    rptSub = new DataDynamics.ActiveReports.ActiveReport();
    //Load the rpx file into the generic report
    System.Xml.XmlTextReader xtr = new
System.Xml.XmlTextReader(this.SubReport1.ReportName);
    rptSub.LoadLayout(xtr);
    xtr.Close();
    //Connect data to the main report
    string connString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Program
Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB;Persist Security Info=False";
    string sqlString = "Select * from orders inner join customers on orders.customerid =
customers.customerid order by CompanyName,OrderDate";
    DataDynamics.ActiveReports.DataSources.OleDBDataSource ds = new
DataDynamics.ActiveReports.DataSources.OleDBDataSource();
    ds.ConnectionString = connString;
    ds.SQL = sqlString;
    rpt.DataSource = ds;
}

```

To write the code to load the xml-based report into the ActiveReports viewer

1. From the Visual Studio toolbox, drag the ActiveReports viewer control onto the Windows Form and set its **Dock** property to **Fill**.
2. Double-click the title bar of the Windows Form containing the viewer to create a Form_Load event.
3. Add code to load the RPX into a generic ActiveReport and display it in the viewer.

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

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the form load event.

```

Dim rpt As New DataDynamics.ActiveReports.ActiveReport()
Dim xtr As New System.Xml.XmlTextReader("C:\MyProjectPath\rptMain.rpx")
rpt.LoadLayout(xtr)
xtr.Close()
rpt.Run()
Viewer1.Document = rpt.Document

```

To write the code in C#

C# code. Paste INSIDE the form load event.

```

DataDynamics.ActiveReports.ActiveReport rpt = new
DataDynamics.ActiveReports.ActiveReport();
System.Xml.XmlTextReader xtr = new
System.Xml.XmlTextReader(@"C:\MyProjectPath\rptMain.rpx");

```

```
rpt.LoadLayout(xtr);
xtr.Close();
rpt.Run();
viewer1.Document = rpt.Document;
```

Creating a Basic End User Report Designer (Pro Edition)

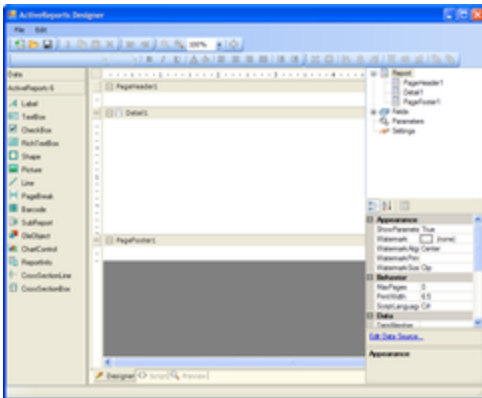
Using ActiveReports Professional Edition, you can set up a custom end-user report designer. This walkthrough illustrates how to set up a basic end-user report designer on a Windows form. (The Designer control is not supported on the Web.)

Note: The **Designer** control requires the .NET Framework full profile version. To ensure you are using the full profile version, go to the Visual Studio **Project Properties**, then to the **Compile** tab, **Advanced Compile Options...** (for Visual Basic projects) or to the **Application** tab (for C# projects) and in the **Target framework** field select a full profile version.

This walkthrough is split up into the following activities:

- Adding controls to the form
- Adding code to import the toolbox library
- Adding an OnExit method
- Adding code to create a data toolbox group
- Adding code to set up the toolbox, menus, tool strips and status bar
- Adding code to display the selected object in the status bar

When you have finished this walkthrough, you will have a working end-user report designer that looks like the following.



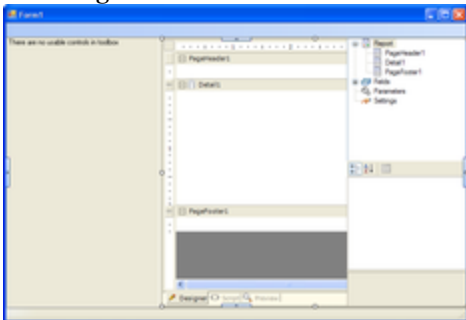
Note: If you need help with adding the Designer controls to your Visual Studio toolbox, see **Adding ActiveReports Controls**.

To add controls to the form

1. Change the Name property of your Windows form to **formDesigner**.
2. Resize the form so that you can comfortably add the controls.
3. Drag the following controls in the order listed from the Visual Studio toolbox onto the form, setting the properties as indicated. (If you have not yet added the ActiveReports controls to your toolbox, see the **Adding ActiveReports Controls** topic.)
Controls for the form

| Control | Parent | Name | Dock | Miscellaneous |
|--------------------|---|---------------------|------|---|
| ToolStripContainer | formDesigner | ToolStripContainer1 | Fill | LeftToolStripPanel Enabled = False; RightToolStripPanel Enabled = False |
| SplitContainer | ToolStripContainer1 | SplitContainer1 | Fill | |
| StatusStrip | ToolStripContainer1 BottomToolStripPanel | arStatus | | |
| Designer | SplitContainer1 Panel2 | arDesigner | None | Anchor = Top, Bottom, Left, Right; Resize and move as necessary. |
| ReportExplorer | SplitContainer1 Panel2 | arReportExplorer | None | ReportDesigner = arDesigner; Anchor = Top, Right; Resize and move as necessary. |
| Toolbox | SplitContainer1 Panel1 | arToolbox | Fill | |
| PropertyGrid | SplitContainer1 Panel2 | arPropertyGrid | None | Anchor = Top, Bottom, Right; Resize and move as necessary. |

4. Select arDesigner and in the Properties window, drop down the **PropertyGrid** property and select **arPropertyGrid**.
5. With the controls added in the correct order and all of the above properties set, the form looks similar to the following:



To import the Toolbox library

1. Right-click the form and select **View Code**.
2. In the code view that appears, add the following code to give your project access to the Toolbox library.

The following examples show what the code looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste ABOVE the formDesigner class.

```
'Visual Basic
'Add the following Imports statements
Imports DataDynamics.ActiveReports
Imports DataDynamics.ActiveReports.Design
```

```
Imports DataDynamics.ActiveReports.Design.Toolbox
```

To write the code in C#

C# code. Paste ABOVE the formDesigner class.

```
//C#
//Add the following using statements
using DataDynamics.ActiveReports;
using DataDynamics.ActiveReports.Design;
using DataDynamics.ActiveReports.Design.Toolbox;
```

To add an OnExit method

1. Right-click in any section of formDesigner, and select **View Code**.
2. In the code view that appears, add the following code to create an OnExit method that you can call from the Exit menu item we create in the next procedure.

The following examples show what the code looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the formDesigner class.

```
'Visual Basic
Private Sub OnExit(ByVal sender As Object, ByVal e As EventArgs)
    Close()
End Sub
```

To write the code in C#

C# code. Paste INSIDE the formDesigner class.

```
//C#
private void OnExit(object sender, EventArgs e)
{
    Close();
}
```

To create a data toolbox group

1. Add the following code right after the OnExit method to create a data group on the toolbox.
2. This code creates a LoadTools method that you can call in the formDesigner Load event to load the new toolbox group into the toolbox.

The following examples show what the code looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the formDesigner class.

```
Private Sub LoadTools(ByVal arToolbox As
DataDynamics.ActiveReports.Design.Toolbox.Toolbox)
    'Add Data Providers
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.DataSet)), "Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.DataView)), "Data")
    Me.arToolbox.AddToolboxItem(New
```



```

System.Drawing.Design.ToolboxItem(GetType(System.Data.OleDb.OleDbConnection)), "Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.OleDb.OleDbDataAdapter)), "Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.Odbc.OdbcConnection)), "Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.Odbc.OdbcDataAdapter)), "Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.SqlClient.SqlConnection)),
"Data")
    Me.arToolbox.AddToolboxItem(New
System.Drawing.Design.ToolboxItem(GetType(System.Data.SqlClient.SqlDataAdapter)),
"Data")
End Sub

```

To write the code in C#

C# code. Paste INSIDE the formDesigner class.

```

private void LoadTools(DataDynamics.ActiveReports.Design.Toolbox.Toolbox arToolbox)
{
    //Add Data Providers
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.DataSet)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.DataView)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.OleDb.OleDbConnection)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.OleDb.OleDbDataAdapter)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.Odbc.OdbcConnection)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.Odbc.OdbcDataAdapter)), "Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.SqlClient.SqlConnection)),
"Data");
    this.arToolbox.AddToolboxItem(new
System.Drawing.Design.ToolboxItem(typeof(System.Data.SqlClient.SqlDataAdapter)),
"Data");
}

```

To set up the designer's toolbox, menus, toolstrips and status bar

1. In the Design view of the form, double-click the title bar of formDesigner. This creates an event-handling method for the formDesigner Load event.
2. Add code to the handler to:
 - Set up the toolbox
 - Set up the menu and tool strips
 - Add an Exit command to the menu
 - Set up the status bar

The following examples show what the code for the method looks like.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the formDesigner Load event.

```
'Note: Assigning the ToolBox to the designer before calling NewReport
' automatically adds the default controls to the toolbox in a group called
' "ActiveReports 6"
LoadTools(arToolbox)
arDesigner.Toolbox = arToolbox

' Add Menu and ToolStrips to Form
Dim menuStrip As ToolStrip = arDesigner.CreateToolStrips(DesignerToolStrips.Menu)(0)
Dim fileMenu As ToolStripDropDownItem = CType(menuStrip.Items(0),
ToolStripDropDownItem)

' Add an Exit command to the File menu
fileMenu.DropDownItems.Add(New ToolStripMenuItem("Exit",
DataDynamics.ActiveReports.Design.Images.Delete, AddressOf OnExit))

Dim panel As ToolStripPanel = ToolStripContainer1.TopToolStripPanel
panel.Join(menuStrip, 0)
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Zoom)(0), 1)
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Undo)(0), 1)
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Edit)(0), 1)
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Report)(0), 1)

panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Layout)(0), 2)
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Format)(0), 2)

'Set up the Status Bar
Dim tsLabel1 As ToolStripStatusLabel = New ToolStripStatusLabel()
tsLabel1.Spring = True
tsLabel1.BorderStyle = Border3DStyle.Sunken
arStatus.Items.Add(tsLabel1)
```

To write the code in C#**C# code. Paste INSIDE the formDesigner Load event.**

```
//C#
//Note: Assigning the ToolBox to the designer before calling NewReport
// automatically adds the default controls to the toolbox in a group called
// "ActiveReports 6"
LoadTools(arToolbox);
arDesigner.Toolbox = arToolbox;

// Add Menu and CommandBar to Form
ToolStrip menuStrip = arDesigner.CreateToolStrips(DesignerToolStrips.Menu)[0];

ToolStripDropDownItem fileMenu = (ToolStripDropDownItem)menuStrip.Items[0];

// Add an Exit command to the File menu
fileMenu.DropDownItems.Add(new ToolStripMenuItem("Exit",
DataDynamics.ActiveReports.Design.Images.Delete, OnExit));

ToolStripPanel panel = ToolStripContainer1.TopToolStripPanel;
panel.Join(menuStrip, 0);
    panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Zoom)[0], 1);
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Undo)[0], 1);
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Edit)[0], 1);
```

```

panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Report)[0], 1);

panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Layout)[0], 2);
panel.Join(arDesigner.CreateToolStrips(DesignerToolStrips.Format)[0], 2);

//Set up the Status Bar
ToolStripStatusLabel tsLabel1 = new ToolStripStatusLabel();
tsLabel1.Spring = true;
tsLabel1.BorderStyle = Border3DStyle.Sunken;
arStatus.Items.Add(tsLabel1);

```

To display the selected object in the status bar

The following examples show what the code to display objects in the status bar looks like.

To write the code in Visual Basic.NET

1. Right-click in any section of formDesigner, and select **View Code** to display the code view for the Windows Form.
2. At the top left of the code view for formDesigner, click the drop-down arrow and select **arDesigner**.
3. At the top right of the code window, click the drop-down arrow and select **SelectionChanged**. This creates an event-handling method for the arDesigner SelectionChanged event.

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

Visual Basic.NET code. Paste INSIDE the SelectionChanged event.

```

'Display the current selection in the designer's status bar
Dim curSelection As String = ""
Dim selectionEnum As IEnumerator = Nothing
If Not (arDesigner.Selection Is Nothing) Then
    selectionEnum = arDesigner.Selection.GetEnumerator()
End If
While Not (selectionEnum Is Nothing) AndAlso selectionEnum.MoveNext()
    If TypeOf selectionEnum.Current Is Section Then
        curSelection = curSelection + (CType(selectionEnum.Current, Section)).Name + ", "
    End If
    If TypeOf selectionEnum.Current Is ARControl Then
        curSelection = curSelection + (CType(selectionEnum.Current, ARControl)).Name + ", "
    End If
    If TypeOf selectionEnum.Current Is Field Then
        curSelection = curSelection + (CType(selectionEnum.Current, Field)).Name + ", "
    End If
    If TypeOf selectionEnum.Current Is Parameter Then
        curSelection = curSelection + (CType(selectionEnum.Current, Parameter)).Key + ", "
    End If
    If TypeOf selectionEnum.Current Is ActiveReport Then
        curSelection = curSelection + (CType(selectionEnum.Current, ActiveReport)).Document.Name + ", "
    End If
End While
If Me.arStatus.Created AndAlso Not (Me.arStatus.Items(0) Is Nothing) Then
    If Not (curSelection = "") Then
        Me.arStatus.Items(0).Text = "Current Selection: " + curSelection.Substring(0, curSelection.Length - 2)
    End If
End If

```

```

Else
    Me.arStatus.Items(0).Text = "No Selection"
End If
End If

```

To write the code in C#

1. Click in the designer control on formDesigner to select **arDesigner**.
2. Click on the events icon in the **Properties** window to display available events for the control.
3. Double-click **SelectionChanged**. This creates an event-handling method for the arDesigner_SelectionChanged event.

C# code. Paste INSIDE the SelectionChanged event.

```

//C#
//This will display the current selection in the designer's status bar
string curSelection = "";
System.Collections.IEnumerator selectionEnum = null;
if(arDesigner.Selection != null)
    selectionEnum = arDesigner.Selection.GetEnumerator();
while(selectionEnum != null && selectionEnum.MoveNext())
{
    if(selectionEnum.Current is Section)
        curSelection = curSelection + (selectionEnum.Current as Section).Name + ", ";
    if(selectionEnum.Current is ARControl)
        curSelection = curSelection + (selectionEnum.Current as ARControl).Name + ", ";
    if(selectionEnum.Current is Field)
        curSelection = curSelection + (selectionEnum.Current as Field).Name + ", ";
    if(selectionEnum.Current is Parameter)
        curSelection = curSelection + (selectionEnum.Current as Parameter).Key + ", ";
    if(selectionEnum.Current is ActiveReport)
        curSelection = curSelection + (selectionEnum.Current as
ActiveReport).Document.Name + ", ";
}
if(this.arStatus.Created && this.arStatus.Items[0] != null)
{
    if(curSelection != "")
        this.arStatus.Items[0].Text = "Current Selection: " + curSelection.Substring(0,
curSelection.Length - 2);
    else
        this.arStatus.Items[0].Text = "No Selection";
}

```

Web Viewer (Pro Edition)

The ActiveReports WebViewer control allows you to easily publish reports to the web for viewing in the browser. The client machine does not require ActiveReports or ASP.NET to be installed. If you use the PDF viewer type, the client machine requires the Adobe Acrobat Reader, and if you use the Flash viewer type, the client machine requires the Adobe Flash Player.

The WebViewer also takes advantage of a report queuing technology to ensure the reports are executed and output efficiently. To use the WebViewer you will select an ActiveReport using the Report property of the WebViewer 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:

- Creating an ASP.NET Web application using ActiveReports
- Connecting the report to a data source
- Setting up a report
- Adding the ActiveReports WebViewer control to the Web Form

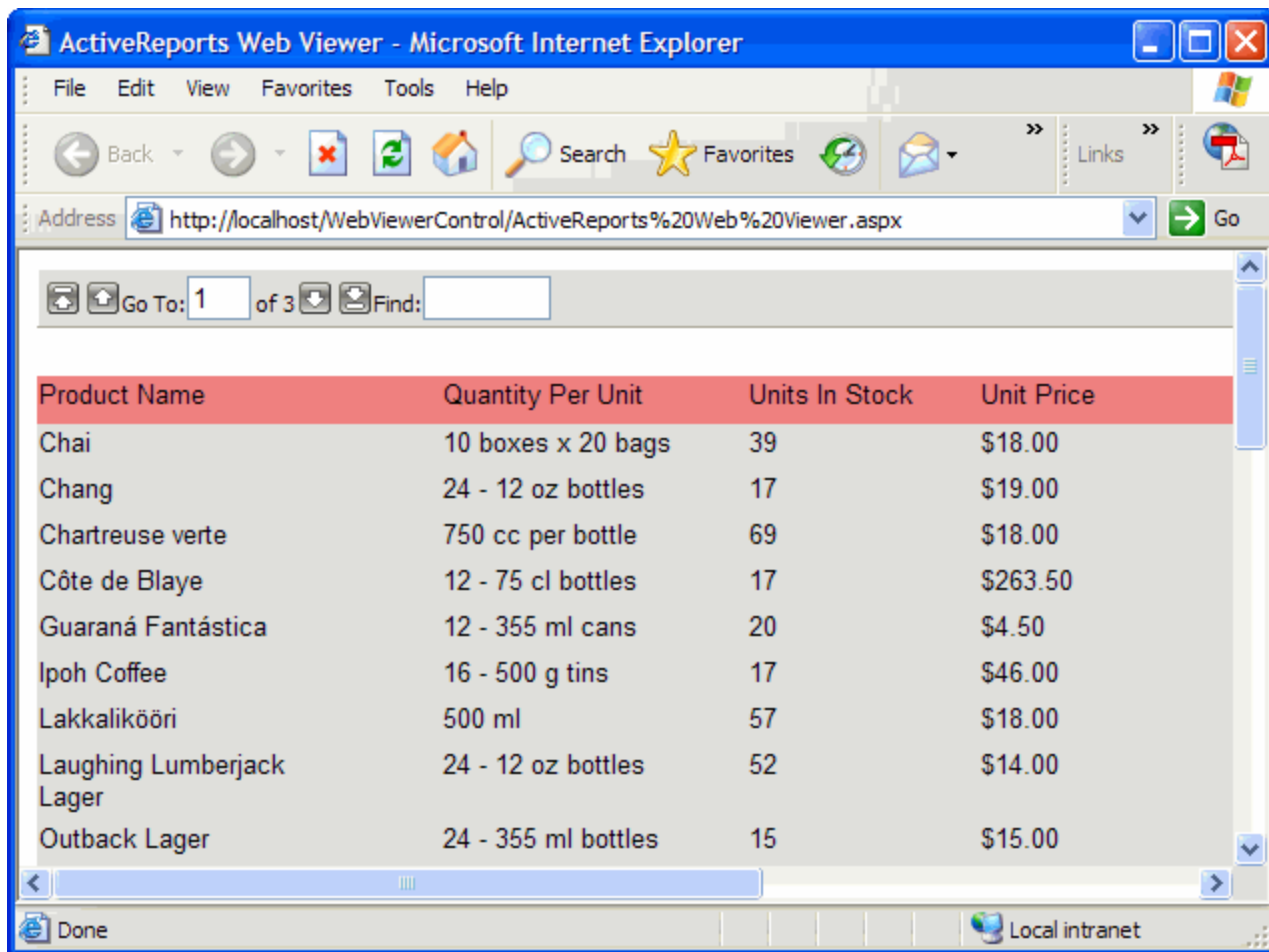
Caution: The WebViewer does not support the use of Ole Objects.

To complete the walkthrough, you must have access to the Northwind database.

A copy is located at C:\Program Files\GrapeCity\ActiveReports 6\Data\NWIND.MDB (on a **64-bit Windows operating system**, a copy is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Data\NWIND.MDB).

You must also have access to Internet Information Services either from your computer or from the server to **Configure the HTTPHandlers**.

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 |

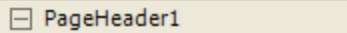
To create an ASP.NET Web application with ActiveReports

1. From the Visual Studio **File** menu, select **New**, then **Project**.
2. In the New Project dialog that appears, select **ASP.NET Web Application** (If you have the **.NET Framework 4**, in the **Visual Studio 2010** New Project dialog that appears, select **ASP.NET Empty Web Site**).
3. Rename the project and click the **OK** button.
4. (For Visual Studio 2010 **.NET Framework 4** only) From the Visual Studio 2010 **Project** menu, select **Add New Item**.

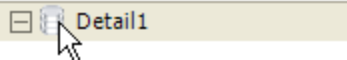
5. (For Visual Studio 2010 .NET Framework 4 only) Select **Web Form**, rename it to **Default.aspx** and click **Add**.
6. (For Visual Studio 2010 .NET Framework 4 only) In the Solution Explorer, right-click the Web Form that you have added and select **Set As Start Page**.
7. From the **Project** menu, select **Add New Item**.
8. Select **ActiveReports 6 (code-based) File**, rename it, and click **Add**.

To connect the report to a data source

1. Click the gray report DataSource icon on the Detail section band to open the Report Data Source dialog.



PageHeader1



Detail1

2. On the "OLE DB" tab, next to Connection String, click the **Build** button.
3. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button.
4. Click the ellipsis (...) button to browse to the Northwind database. Click **Open** once you have selected the appropriate access path.
5. Click **OK** to close the window and fill in the Connection String field.
6. In the Query field, enter the following SQL query

SQL Query

```
SELECT * FROM Products ORDER BY CategoryID, ProductName
```

7. Click **OK** to save the data source and return to the report design surface.

To set up the report

1. Right-click the design surface of the report and choose **Insert**, then **GroupHeader/Footer** to add a group header and footer section.
2. Make the following changes to the group header:
 - Change the **BackColor** property to **PaleVioletRed**.
 - Change the **DataField** property to **CategoryID**.
 - Change the **GroupKeepTogether** property to **FirstDetail**.
 - Change the **KeepTogether** property to **True**.
3. Add the following controls to the GroupHeader section:
Group header controls

| Control | Size | Text | Location |
|---------|---------------|-------------------|----------|
| Label | 1, 0.198 in | Product Name | 0, 0 |
| Label | 1.1, 0.198 in | Quantity Per Unit | 2.1, 0 |
| Label | 1, 0.198 in | Units In Stock | 3.7, 0 |
| Label | 1, 0.198 in | Unit Price | 4.9, 0 |

4. Select the detail section, and in the Properties window, make the following changes:
 - Change the **CanShrink** property to **True**.
 - Change the **BackColor** property to **LightGray**.
5. In the Report Explorer, expand the **Fields** node, then the **Bound** node.


6. Drag the following fields onto the detail section and set the properties of each textbox as indicated.

Detail section fields

| Field | Size | Location | OutputFormat |
|-----------------|-------------|----------|--------------|
| ProductName | 1.8, 0.2 in | 0, 0 | NA |
| QuantityPerUnit | 1.5, 0.2 in | 2.1, 0 | NA |
| UnitsInStock | 1, 0.2 in | 3.7, 0 | NA |
| UnitPrice | 1, 0.2 in | 4.9, 0 | Currency |

To add the ActiveReports WebViewer control to the ASPX

1. On the Default.aspx page, click the **Design** tab at the bottom.
2. From the Toolbox, drag the **WebViewer** control onto the page.

 If you get a **Could not load file or assembly...** message, an old version of ActiveReports was installed when the WebViewer was last added to the toolbox. To add the latest version of the control, right-click the toolbox and select **Choose Items**. In the Choose Toolbox Items dialog, clear the WebViewer check box and select the WebViewer check box with the highest version number in the Assembly Name column.

3. Drag the bottom right corner of the control to enlarge the viewer.
4. In the Properties window, drop down the **ReportName** property and select your report.
5. Drop down the ViewerType property, and select from the following:
 - **HtmlViewer** (default) displays the report in an HTML version of the report viewer, with page navigation and a **Find** function.
 - **RawHtml** displays the report as one long HTML page with no viewer interface.
 - **AcrobatReader** displays the report in the Adobe Reader. (The user must have the Adobe Reader installed.)
 - **FlashViewer** displays the report in a Flash version of the report viewer.

 **Important:** To use the Flash Viewer, copy the **ActiveReports.FlashViewer.swf** file into your project folder.

This file is located in C:\Program Files\GrapeCity\ActiveReports 6\Deployment (on a **64-bit Windows operating system**, this file is located in C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment).


6. Run the project to display the report in the selected viewer.

Flash Viewer

The new FlashViewer is interactive and customizable.

This walkthrough is split up into the following activities:

- Creating an ASP.NET Web site using ActiveReports
- Adding the ActiveReports WebViewer control to the aspx page
- Setting up the FlashViewer

 **Caution:** The WebViewer does not support the use of Ole Objects.

To complete the walkthrough, you must have access to Internet Information Services either from your computer or from the server to **Configure the HTTPHandlers**.

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

NorthWind Traders
One Portals Way
Twin Points WA 98156

Invoice
 Order #: 10248
 Order Date: 08/04/1994

Bill To
 Vins et alcools Chevalier
 59 rue de l'Abbaye
 Reims
 51100 France

Ship To
 Vins et alcools Chevalier
 59 rue de l'Abbaye
 Reims
 51100 France

Sales Person: Steven Buchanan
Shipped: 8/16/1994
Via: Federal Shipping

| Product ID | Product Name | Qty | Unit Price | Discount | Total |
|------------|-------------------------------|-----|------------|----------|----------|
| 42 | Singaporean Hokkien Fried Mee | 10 | \$9.80 | | \$98.00 |
| 72 | Mozzarella di Giovanni | 5 | \$34.80 | | \$174.00 |
| 11 | Queso Cabrales | 12 | \$14.00 | | \$168.00 |

To create an ASP.NET Web site with ActiveReports

1. From the Visual Studio **File** menu, select **New Web Site...** (From the **Visual Studio 2010 File** menu, select **New**, then **Web Site...**).
2. In the New Web Site dialog that appears, select **ASP.NET Web Site**. (In the **Visual Studio 2010** New Web Site dialog that appears, select **ASP.NET Empty Web Site**).
3. Rename the website and click the **OK** button.
4. (For Visual Studio 2010 only) From the **Visual Studio 2010 Website** menu, select **Add New Item**.
5. (For Visual Studio 2010 only) Select **Web Form** and click **Add**.
6. From the **Website** menu, select **Add New Item**.
7. Select **ActiveReports 6 (code-based) File**, rename it, and click **Add**.
8. In the Microsoft Visual Studio message box that appears, click **Yes** to place the report inside the App_Code folder so that it is generally consumable in your site. The code view of the report appears.
9. In the Solution Explorer, expand the **App_Code** folder, right-click your report and select **View Designer**. This displays the design surface where you can add controls to the report and set its properties.

Tip: For help with this, see the **Create Common Reports** topics.

To add the WebViewer control to the aspx page

1. On the Default.aspx page, click the **Design** tab at the bottom.
2. From the Toolbox, drag the **WebViewer** control onto the page.

Tip: If you get a **Could not load file or assembly...** message, an old version of ActiveReports was installed when the WebViewer was last added to the toolbox. To add the latest version of the control, right-click the toolbox and select **Choose Items**. In the Choose Toolbox Items dialog, clear the old WebViewer


check box and select the WebViewer check box with the highest version number.

3. Drag the bottom right corner of the control to enlarge the viewer to at least 740 px wide and 1000 px high to avoid horizontal and vertical scrollbars at run time.
4. In the Properties window, drop down the **ReportName** property and select your report.
5. Drop down the ViewerType property, and select **FlashViewer**.

To set up the FlashViewer

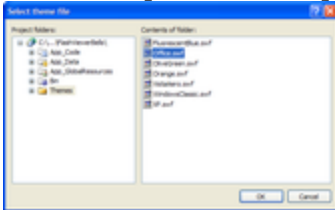
1. In Windows Explorer, navigate to **C:\Program Files\GrapeCity\ActiveReports 6\Deployment\Flash** (on a **64-bit Windows operating system**, navigate to **C:\Program Files (x86)\GrapeCity\ActiveReports 6\Deployment\Flash**).
2. In the Flash folder, copy the **ActiveReports.FlashViewer.swf** file and the **Themes** folder.

 The **ActiveReports.FlashViewer.Resources.swf** file is only used for localization.

3. Paste the swf file and Themes folder into your main project folder.
4. Back in Visual Studio, in the Solution Explorer, click the  **Refresh** button to see the swf file and Themes folder.
5. Click the WebViewer so that it is selected in the Properties window.
6. Expand the **FlashViewerOptions** property node, and click the **ThemeUrl** property to show the ellipsis button.



7. Click the ellipsis button to open the **Select theme file** dialog.



8. In the **Project folders** pane to the left, select the **Themes** folder. The included themes display in the **Contents of folder** pane to the right.
9. Select a theme and click **OK**.
10. Run the project to view the results.

Customizing the Flash Viewer UI

You can customize the Flash Viewer UI, using javascript functions. This walkthrough illustrates how to customize the look and feel of the Flash Viewer.

The walkthrough is split up into the following activities:

- Adding code to customize the Flash Viewer options
- Adding code to handle the Flash Viewer events

To follow the steps of this walkthrough, you must first complete the steps on creating an ASP.NET Web site and setting up the FlashViewer, described in the **Flash Viewer** walkthrough.

To customize the Flash Viewer options

1. Select the WebViewer on your .aspx page, and in the Properties window, expand the **FlashViewerOptions** node.
2. Set the property **UseClientApi** (FlashViewerOptions) to True (this property is set to False by default).



3. Declare a variable and attach the FlashViewer by adding the following code onto the .aspx **Source** view.

Paste the code into the Source view of .aspx inside the <head> and </head> tags

```
<script language="javascript" type="text/javascript">
var viewer;

function init() {
  DataDynamics.ActiveReport.Viewer.OnLoad ("WebViewer1", function() {
    viewer = DataDynamics.ActiveReport.Viewer.Attach("WebViewer1");
  });
}
</script>
```

Replace the <body> tag on the Source view of .aspx

```
<body onload="return init()">
```

4. Drag the **Html Input (button)** control from the Visual Studio toolbox onto the .aspx design view, containing the WebViewer control.
5. Double-click the **Button** control and paste the following code:

Paste the code into the Source view of .aspx inside the <script> and </script> tags


```
function Button1_onclick() {
  var zoom = viewer.getZoom();
  alert("Check that zoom property is changed from " + zoom);
  zoom += 0.1;
  viewer.setZoom(zoom);
  alert("to " + zoom);
}
```

 **Note:** You can declare a variable and attach the FlashViewer in the event handler directly. This would replace the actions in the step 3 above.

Paste the code into the Source view of .aspx inside the <script> and </script> tags

```
function Button1_onclick()
{
  var viewer;
  viewer = DataDynamics.ActiveReport.Viewer.Attach("WebViewer1");
  ...
}
```

Available Flash Viewer properties

 The properties below use a pair of the getter and setter functions, which names are formed as {get|set}<PropertyName>().

| Property Name | Description |
|-------------------------------|--|
| CurrentPage | Integer. Gets or sets the page for a currently selected view. |
| HyperLinkBackColor | String. Gets or sets the background color for all hyperlinks in a document. This value is applied only after a new document is loaded. The format is "#FF0000" (#RRGGBB). |
| HyperLinkForeColor | String. Gets or sets the color for all hyperlinks in a document. This value is applied only after a new document is loaded. The format is "#FF0000" (#RRGGBB). |
| HyperLinkUnderline | Boolean. Gets or sets a value determining whether the text of all hyperlinks in a document is underlined. This value is applied only after a new document is loaded. |
| SearchResultsBackColor | String. Gets or sets the background color of the highlighted text when using the Find dialog. The format is "#FF0000" (#RRGGBB). |
| SearchResultsForeColor | String. Gets or sets the color of the highlighted text when using the Find dialog. The format is "#FF0000" (#RRGGBB). |
| PaperColor | String. Gets or sets the paper color of the report. The format is "#FF0000" (#RRGGBB). |
| ShowSplitter | Boolean. Gets or sets a value determining whether to display a splitter. If set to True, the splitter will be shown. |
| TargetView | String. Gets or sets a value that specifies which view - Primary or Secondary, is currently active. If the ShowSplitter property is set to False, then you cannot switch to the Secondary view. |
| ThemeUrl | String. Gets or sets a value specifying the relative URL of a skin to use on the Flash Viewer. |
| Zoom | Integer. Gets or sets a value that specifies the zoom level at which to display the report. |
| TocPanel | Object. Gets the Table of Contents Panel object. This property is read-only. <ul style="list-style-type: none"> • RightAlign Allows to align the TOC panel to the right of the Viewer. • ThumbnailsVisible Allows to specify whether to display a pane with thumbnail views of pages. • TocVisible Allows to specify whether to display the table of contents pane. • Visible Allows to specify whether to display the Table of contents. • Width Allows to specify the Table of contents width. |
| ViewType | String. Gets or sets the current ViewType value. |
| EventsHandler | Object. Gets or sets the events handler container. |

Available Flash Viewer methods

| Method Name | Description |
|--|--|
| LoadDocument (string documentUrl) | Loads the RDF document. You may use an RDF file that resides on a server, RpxHandler, CompiledReportHandler or custom Http handler implementation to provide a streamed document. This method cancels the current document's |

| | |
|---|---|
| | loading. |
| CancelDocumentLoad() | Cancels the loading of a current document. |
| Print(PrintOptions options) | Causes the Viewer to wait until all required pages are loaded, displayed in the Print dialog and starts printing. The PrintOptions are similar to the <code>WebViewer.PrintOptions</code> , except that the <code>PageRangesCollection</code> methods are merged into the <code>PrintOptions</code> class. |
| CreatePrintOptions() | Creates options for the Print() method. |
| setViewType(string viewType, int multiPageRows, int multiPageCols) | Specifies the view mode. Possible values for the first parameter are specified in the ViewType enumeration. The last two parameters are applied for ViewType.MultiPage only. |

To handle Flash Viewer Events


1. Declare a variable, attach the Flash Viewer and add event handlers by adding the following code onto the **.aspx Source** view.

Paste the code into the Source view of .aspx inside the <head> and </head> tags

```
<script language="javascript" type="text/javascript">
var viewer;
function init() {
DataDynamics.ActiveReport.Viewer.OnLoad("WebViewer1", function() {
viewer = DataDynamics.ActiveReport.Viewer.Attach("WebViewer1");
viewer.setEventsHandler({
  OnLinkClick: function(e) {
    alert(e.Link); //specifies url of the link item, string
    return true;
  },
  OnError: function(e) {
    alert(e.Message); //error message, string
    alert(e.ErrorType); //possible types are "Error" and "Warning", string
    return false;
  }
});
});
}
</script>
```

Replace the <body> tag on the Source view of .aspx

```
<body onload="return init()">
```

 **Note:** <EVENTS> are described in detail in the **Available events** list below.

Available events

Event

OnLinkClick(LinkEventArgs)

Description

Specifies the URL value of a linked item or a string. This event is raised when a report object with a hyperlink is clicked; this event allows to override the

default Hyperlinks behavior that simply opens another browser window. Cancelable.

The event handler receives an argument of type `LinkEventArgs` containing data related to this event. The handler argument has the field "Link". The field "Link" returns the hyperlink URL value.

Code example

```
OnLinkClick: function(e)
{
    alert(e.Link); //specifies url of the
    link item, string
    return true;
}
```

OnError(ErrorEventArgs)

Fires when an application fires an error or a warning. Use this event to customize FlashViewer error messages and warnings.

The event handler receives an argument of type `ErrorEventArgs` containing data related to this event. The handler argument has the following fields - "ErrorType" and "Message".

The field "ErrorType" contains the error type value; the field "Message" contains a description of a Flash Viewer problem.

This event allows suppressing any notifications to a user.

Code example

```
OnError: function(e)
{
    alert(e.Message);
    //error message, string
    alert(e.ErrorType);
    //possible types are "Error" and
    "Warning", string
    return false;
}
```

OnLoadProgress(LoadProgressArgs)

Raised during the processing of a report.

The event handler receives an argument of type `LoadProgressArgs` containing data related to this event. The handler argument has the following fields - "PageCount", "PageNumber" and "State".

The field "PageCount" returns the total count of report pages.

The field "PageNumber" returns the page number of the processed report.

The field "State" returns the value, indicating the state of the report's processing; the possible values are

"Completed", "InProgress" and "Cancelled".

Code example

```

OnLoadProgress: function(e)
{
    if(e.State == "InProgress") {
        if(e.PageNumber == 10)
        {
            alert("10 pages
are loaded");
        }
    }

    if(e.State == "Cancelled"){
        alert("Report
processing is cancelled");
    }

    if(e.State == "Completed")//
possible value are Completed,
InProgress and Cancelled
    {
        alert("Report loading
is completed, total page count is" +
e.PageCount);
    }

    //return value is ignored in
this event
}

```

OnTargetViewChanging(TargetViewChangeEventArgs)

Raised while a report's view is changing.

The event handler receives an argument of type TargetViewChangeEventArgs containing data related to this event. The handler argument has the following fields - "CurrentView" and "NewView".

The field "CurrentView" returns the currently selected view value.

The field "NewView" returns the newly selected view value.

Code example

```

OnTargetViewChanging: function(e)
{
    alert("Currently
selected view is " + e.CurrentView);
//gets currently selected view, string
    alert("Newly selected
view is " + e.NewView);
//gets newly selected view, string
    return false;
//cancelable event
}

```

OnToolClick(ToolClickEventArgs)

Raised by clicking the toolbar button. Cancelable.

The event handler receives an argument of type `ToolClickEventArgs` containing data related to this event. The handler argument has the following field - "Tool".

The field "Tool" returns the name of a clicked button.

Code example


```
OnToolClick: function(e)
{
    alert(e.Tool);
    return false;
}
```

OnCurrentPageChanged(CurrentPageChangeEventArgs)

Raised each time a current page is changed within the current view programmatically or by any user navigation command. This event is also raised when the current view (primary or secondary) is changed.

The event handler receives an argument of type `CurrentPageChangedEventArgs` containing data related to this event. The handler argument has the following fields - "PageNumber" and "ViaApi".

The field "PageNumber" contains the 1-based page number. The field "ViaApi" specifies whether the event is raised by setting the page number in the **CurrentPage** property.

 **Note:** Use the "return true;" value to show that the client side has handled the event. The "return false;" value indicates that the event was not handled.

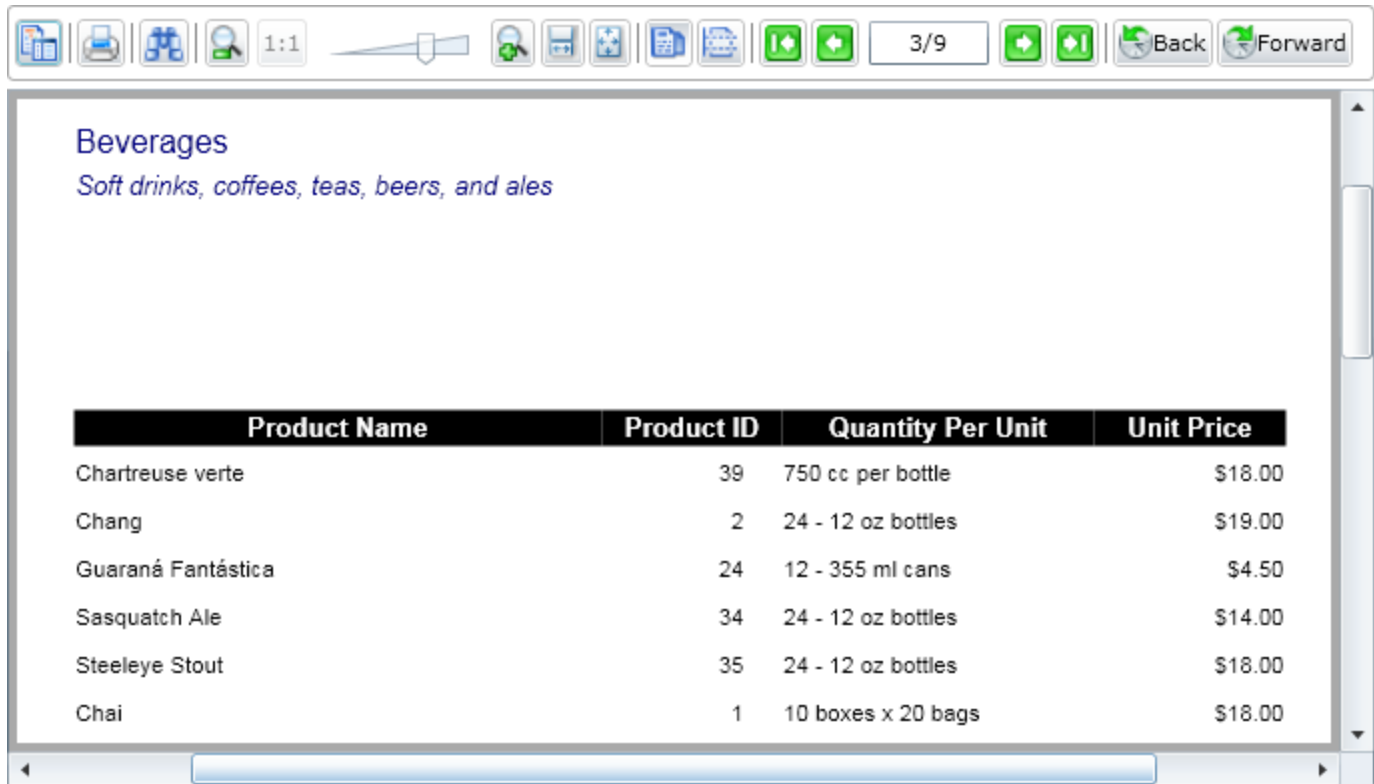
Silverlight Viewer (Pro Edition)

This walkthrough introduces you to Silverlight in ActiveReports and will guide you through the process of creating a Silverlight application in Visual Studio and viewing a report in the ActiveReports Silverlight Viewer.

This walkthrough is split up into the following activities:

- Creating a Silverlight application in Visual Studio 2010
- Adding the ActiveReports Silverlight Viewer control to the xaml page
- Binding a report to the ActiveReports Silverlight Viewer
- Previewing a report
- Printing a report

When you have completed this walkthrough, you will have a Web site that looks similar to the following:

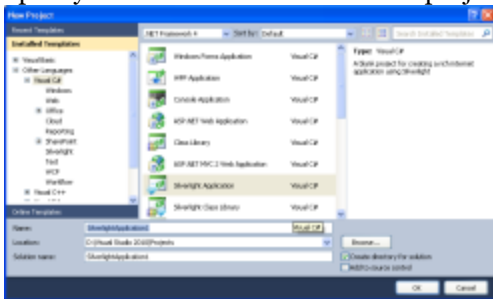


Beverages
Soft drinks, coffees, teas, beers, and ales

| Product Name | Product ID | Quantity Per Unit | Unit Price |
|--------------------|------------|--------------------|------------|
| Chartreuse verte | 39 | 750 cc per bottle | \$18.00 |
| Chang | 2 | 24 - 12 oz bottles | \$19.00 |
| Guaraná Fantástica | 24 | 12 - 355 ml cans | \$4.50 |
| Sasquatch Ale | 34 | 24 - 12 oz bottles | \$14.00 |
| Steeleye Stout | 35 | 24 - 12 oz bottles | \$18.00 |
| Chai | 1 | 10 boxes x 20 bags | \$18.00 |

To create a Silverlight application in Visual Studio 2010

1. On the Visual Studio **File** menu, click **New Project**.
2. In the **New Project** dialog that appears, select **Silverlight Application** in the list of templates.
3. Specify the name and location of the project and then click the **OK** button.



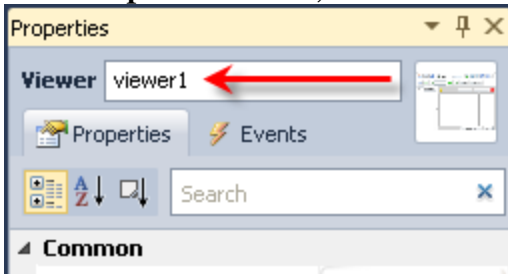
4. In the **New Silverlight Application** dialog, select a method for hosting your Silverlight application and then click the **OK** button.




To add the Silverlight Viewer control

1. Open MainPage.xaml in Solution Explorer.

2. From the **Toolbox**, drag the **Viewer** control to Design view of MainPage.xaml.
3. Drag any corner of the control to enlarge the viewer.
4. In the **Properties** window, rename the Viewer control to **viewer1**.



 **Note:** Microsoft Silverlight 4 Tools is required for the proper functioning of the ActiveReports Silverlight Viewer.

To bind a report to the ActiveReports Silverlight Viewer

To bind a report stream (aspx page)

1. In Solution Explorer, select the *YourProject.Web* directory.
2. On the Visual Studio **Project** menu, click **Add New item**.
3. From the basic list of templates, select **ActiveReports 6 (code-based) File** and create a report as described in the **Basic Data Bound Reports** walkthrough.
4. On the Visual Studio **Project** menu, click **Add New item**.
5. In the dialog that appears, select **Web Form**.
6. Rename the file to **GetReport.aspx** and then click **Add**.
7. Double-click the design view of GetReport.aspx to create an event-handling method for the Page_Load event.
8. On GetReport.aspx.cs/vb that opens, add the following code:

Paste to GetReport.aspx.cs to the Page_Load event

```
YourReportName my_rpt = new YourReportName();

my_rpt.Run();

System.IO.MemoryStream ms = new System.IO.MemoryStream();

my_rpt.Document.Save(ms);

Response.BinaryWrite(ms.ToArray());

Response.End();
```

9. In Design view of MainPage.xaml, select the Viewer control.
10. In the **Properties** window, go to the **Events** view.
11. In the list of events, double-click the viewer1_Loaded event.
12. On MainPage.xaml.cs/vb, add the following code to the viewer1_Loaded event:

Paste to MainPage.xaml.cs to the viewer1_Loaded event

```
this.viewer1.ViewModel.ReportPath = "../GetReport.aspx";
```

To bind an RPX report handler

1. In Solution Explorer, select the *YourProject.Web* directory.
2. On the Visual Studio **Project** menu, click **Add Reference**.
3. In the dialog that appears, go to .NET tab, locate and select **GrapeCity ActiveReports Web Component** and then click **OK**.
4. With the *YourProject.Web* directory selected, on the Visual Studio **Project** menu, click **Add Existing item**.
5. In the dialog that appears, locate and select an RPX report and then click **OK**.
6. On MainPage.xaml, add the following code to Design view:

Paste to the UserControl section on the Design view of MainPage.xaml

```
xmlns:vvm="clr-
namespace:DataDynamics.ActiveReports.ViewModel;assembly=ActiveReports.Silverlight"
```

7. On MainPage.xaml, replace the code with the following one:

Paste to the Design view of MainPage.xaml

```
d:DesignHeight="300" d:DesignWidth="400" xmlns:my="clr-
namespace:DataDynamics.ActiveReports;assembly=ActiveReports.Silverlight"
Loaded="viewer1_Loaded">
```

8. In the **Properties** window, go to the **Events** view.
9. In the list of events, double-click the viewer1_Loaded event.
10. On MainPage.xaml.cs/vb that opens, add the following code to the viewer1_Loaded event:

Paste to MainPage.xaml.cs to the viewer1_Loaded event

```
var relPath = new Uri("../YourReportName.rpx?OutputFormat=Rdf3",
UriKind.RelativeOrAbsolute);
viewer1.ViewModel.LoadDocument.Execute(relPath);
```


11. In Solution Explorer, open the Web.config file.
12. Add the following code to the Web.config file to set an rpx handler. You can find more information on how to configure http handlers in **Configure HTTP Handlers (Pro Edition)** and **Add Report Links to Web Forms (Pro Edition)**.

XML code. Paste in the XML view of the Web.config file inside the system.web section.

```
<httpHandlers>

<add verb="*" path="*.rpx"
type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
Version=6.0.865.0,
Culture=neutral, PublicKeyToken=cc496777c49a3ff" />

</httpHandlers>
```

 **Note:** You should update the Version and PublicKeyToken values to reflect the current version of ActiveReports installed on your machine.

You can find the Version and PublicKeyToken values in the Global Assembly Cache (GAC),
C:\WINDOWS\ASSEMBLY.

To bind an RDF report

1. In Solution Explorer, select the *YourProject.Web* directory.
2. On the Visual Studio **Project** menu, click **Add Existing item**.
3. In the dialog that appears, locate and select an RDF report and click **OK**. You can find sample RDF reports in the

C:\My Documents\GrapeCity\ActiveReports 6\Samples\VB\RDFViewer\RDFs folder.

- On MainPage.xaml, add the following code to Design view:

Paste to the UserControl section on the Design view of MainPage.xaml

```
xmlns:vvm="clr-
namespace:DataDynamics.ActiveReports.ViewModel;assembly=ActiveReports.Silverlight"
```

- On MainPage.xaml, add the following code to the Grid section:

Paste to the Design view of MainPage.xaml to the Grid section

```
<Grid x:Name="LayoutRoot" Background="White">

    <my:Viewer HorizontalAlignment="Left" Name="viewer1" VerticalAlignment="Top"
    Height="300" Width="400">

        <my:Viewer.ViewModel>

            <vvm:ViewerViewModel ReportPath="..\YourReportName.rdf" />

        </my:Viewer.ViewModel>

    </my:Viewer>
</Grid>
```

To bind an RDF report by using the Viewer.ViewModel.ReportPath property

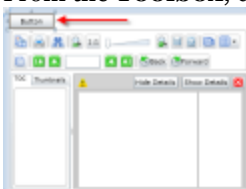
- In Solution Explorer, select the *YourProject.Web* directory.
- On the Visual Studio **Project** menu, click **Add Existing item**.
- In the dialog that appears, locate and select an RDF report and click **OK**. You can find sample RDF reports in the C:\My Documents\GrapeCity\ActiveReports 6\Samples\VB\RDFViewer\RDFs folder).
- In Design view of MainPage.xaml, select the Viewer control.
- In the **Properties** window, go to the **Events** view.
- In the list of events, double-click the viewer1_Loaded event.
- On MainPage.xaml.cs/vb that opens, add the following code to the viewer1_Loaded event:

Paste to MainPage.xaml.cs to the viewer1_Loaded event

```
viewer1.ViewModel.ReportPath = "../YourReportName.rdf";
```

To bind an RDF report by using OpenFileDialog

- In Solution Explorer, select the *YourProject.Web* directory.
- On the Visual Studio **Project** menu, click **Add Existing item**.
- In the dialog that appears, locate and select an RDF report and click **OK**. You can find sample RDF reports in the C:\My Documents\GrapeCity\ActiveReports 6\Samples\VB\RDFViewer\RDFs folder.
- From the **Toolbox**, drag the **Button** control onto the Viewer control in Design view of MainPage.xaml.





5. In the **Properties** window, rename the Button control to **btnOpenReport**.
6. In the list of events, double-click the Click event.
7. On MainPage.xaml.cs/vb that opens, add the following code to the btnOpenReport_Click event:

Paste to MainPage.xaml.cs to the btnOpenReport_Click event

```

OpenFileDialog dialog = new OpenFileDialog();

//Show the dialog

bool? dialogResult = dialog.ShowDialog();

if (dialogResult != true) return;

using (System.IO.Stream fs = (System.IO.Stream)dialog.File.OpenRead())

{

    viewer1.ViewModel.LoadDocument.Execute(fs);

}

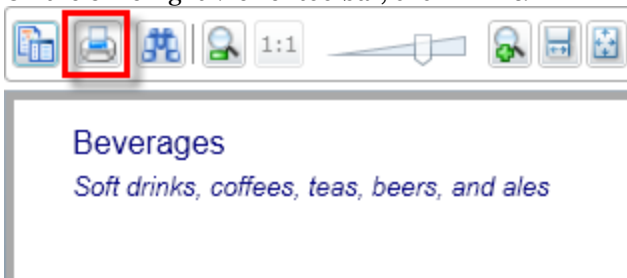
```

To preview a report

1. Press F5 to run the project. The page with the Silverlight Viewer displaying a report, appears in your browser.

To print a report

1. On the Silverlight Viewer toolbar, click **Print**.



2. In the Print dialog that appears, select the printer settings and click **Print**.

Troubleshooting

If you run into an issue while using ActiveReports, you will probably find the solution within this section. Click any short description below to drop down the symptoms, cause, and solution. Or click a link to another section of the troubleshooting guide.

General Troubleshooting

Copy icon missing from the viewer

Symptoms: The copy icon is not showing in the viewer.

Cause: The ActiveReports RTF and Text export filters are not referenced in the project. The viewer has intentionally been designed not to require the export filters so no extra files are required in distribution.

Solution:

1. In the Solution Explorer, right click References and choose Add Reference.
2. Select GrapeCity ActiveReports Rich Text Format (RTF) Export Component and GrapeCity ActiveReports Text Export Component and click OK.

Errors after installing a new build

Symptoms: When you open a project created with a previous build of ActiveReports after installing a new build, there are errors related to being unable to find the previous build.

Cause: Visual Studio has a property on references called Specific Version. If this property is set to True, the project looks for the specific version that you had installed when you created the report, and throws errors when it cannot find it.

Solution: For each of the ActiveReports references in the Solution Explorer, select the reference and change the Specific Version property to False in the Properties Window.

Blank pages printed between pages or red line in the viewer

Symptoms: Blank pages are printed between pages of the report.

Cause: This problem occurs when the PrintWidth plus the left and right margins exceeds the paper width. For example, if the paper size were set to A4, the PrintWidth plus the left and right margins should not exceed 8.27"; otherwise blank pages will be printed. At run time, ActiveReports marks a page overflow by displaying a red line in the viewer at the position in which the breach has occurred.

Solution: The PrintWidth can be adjusted in the report designer using either the property grid or by dragging the right edge of the report. Page margins, height, and width can be adjusted either through the print properties dialog box in the Report menu under Settings or programmatically in the Report_Start event.

Copying reports results in stacked controls

Symptoms: A report file copied into a new project has all of its controls piled up at location 0, 0.

Cause: The report has become disconnected from its resource file. When you set a report's Localizable property to True, the Size and Location properties of the report's controls are moved to the associated *.resx file, so if you copy or move the report, you must move the *.resx file along with it.

Solution: When you copy a report's *.vb or *.cs file from one project's App_Code folder into the App_Code folder of a new project, you need to also copy its *.resx file from the original project's App_GlobalResources folder into the new project's App_GlobalResources folder.

The project does not work if Integrated Managed Pipeline Mode is enabled

Symptoms: The web project does not work in the application pool if Integrated Managed Pipeline Mode is enabled.

Cause: The application configuration is incorrect for being used in Integrated mode.

Solution: You need to migrate the application configuration. Here is a sample of the command line:

Other code. Paste INSIDE the ReportEnd event.

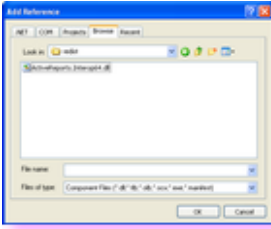
```
"%SystemRoot%\system32\inetsrv\appcmd migrate config YourWebSite/"
```

ActiveReports.Interop64.dll is not available in the default "Add Reference" dialog

Symptoms: ActiveReports.Interop64.dll is not available in the default "Add Reference" dialog.

Cause: ActiveReports.Interop64.dll is located in another folder.

Solution: ActiveReports.Interop64.dll can be found at **C:\Program Files\Common Files\GrapeCity\ActiveReports 6\Redist** (on a **64-bit Windows operating system**, at **C:\Program Files (x86)\Common Files\GrapeCity\ActiveReports 6\Redist**).



A warning message appears when opening a report with the OleObject control

Symptoms: A warning message is displayed when you open a report that contains the OleObject control.

Cause: This problem occurs in case the Microsoft .NET Framework 4.0 Client Profile or .NET Framework 4.0 Full Profile is used.

Solution: You should open the app.config file and set the **useLegacyV2RuntimeActivationPolicy** attribute to true.

XML code. Paste INSIDE the app.config file.

```
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="<WHATEVER>" />
  </startup>
</configuration>
```

The DocumentLoadException occurs when deploying a report with an RDF file on the Windows Azure Developer Portal

Symptoms: The DocumentLoadException occurs when deploying a report with an RDF file on the Windows Azure Developer Portal.

Cause: An RDF file is not copied to the server at the report deployment.

Solution: In Solution Explorer, you should select an RDF file and set its **BuildAction** property to **Content** and its **Copy to Output Directory** property to **Copy always**.

The designer is not refreshed after the PrintWidth property is changed via code

Symptoms: The designer is not refreshed after the PrintWidth property is changed via the following code:

Visual Basic.NET code.

```
Me.arDesigner.Report.PrintWidth = 5.0F
me.arDesigner.Refresh()
```

C# code.

```
this.arDesigner.Report.PrintWidth = 5.0f;
this.arDesigner.Refresh();
```

Cause: The PrintWidth property requires the OnComponentChanged event to refresh the designer layout.

Solution: To change the PrintWidth property, you can use the **UpdateComponent method** or you can use the **PropertyDescriptor.SetValue()** to fire the OnComponentChanged event.

To use the UpdateComponent method as in the following example

Visual Basic.NET code.

```
Me.arDesigner.UpdateComponent(Me.arDesigner.Report, "PrintWidth", 5.0F)
```

C# code.

```
this.arDesigner.UpdateComponent(this.arDesigner.Report, "PrintWidth", 5.0f);
```

To use the `PropertyDescriptor.SetValue()` as in the following example

Visual Basic.NET code.

```
Dim p As System.ComponentModel.PropertyDescriptor =
System.ComponentModel.TypeDescriptor.GetProperties(Me.arDesigner.Report).Item("PrintWidth")
p.SetValue(Me.arDesigner.Report, 5.0F)
```

C# code.

```
System.ComponentModel.PropertyDescriptor p =
System.ComponentModel.TypeDescriptor.GetProperties(this.arDesigner.Report) ["PrintWidth"];
p.SetValue(this.arDesigner.Report, 5.0f);
```


The "Invalid Resx file..." error occurs when rebuilding an ActiveReports application with a watermark in the SP2 or earlier builds

Symptoms: The "Invalid Resx file. Could not load type DataDynamics.ActiveReports.WatermarkDataStream..." error occurs in the report's *.resx file.

Cause: A new private **WatermarkData** property has been added to a report for serializing the **Watermark** property.

Solution:

1. In Solution Explorer, double-click the *YourReport.resx* file to open it in the Managed Resources editor.

 **Note:** If you cannot see the *YourReport.resx* file, click the **Show All Files** button in Solution Explorer.

2. In the Managed Resources editor, select the row **\$this.WatermarkData**.
3. Right-click the selected row and select **Remove Row**.
4. In the dialog that appears, click **Yes**.
5. In Solution Explorer, double-click the *YourReport.Designer* file to open it.
6. In the view that opens, remove the following line:

Visual Basic.NET code.

```
Me.WatermarkData = CType(resources.GetObject("$this.WatermarkData"),
System.IO.Stream)
```

C# code. Remove from the code view of the Windows form

```
this.WatermarkData = ((System.IO.Stream)
(resources.GetObject("$this.WatermarkData")));
```

7. In Solution Explorer, double-click the *YourReport* file to open the Design view of the report.
8. In the Properties window, specify the **Watermark** property again.

The report is not associated with the Visual Studio Designer when adding ActiveReports to a TFS-bound project


Symptoms: When adding ActiveReports to the web site project that is bound to TFS, the report is not associated with the Visual Studio Designer. This situation only occurs in a web site project, where ActiveReports is added to the **App_Code** folder.

Cause: The **FileAttributes.xml** file contains the attribute information for the ActiveReports file that is treated as a component, associated with the Designer. The **FileAttributes.xml** file is loaded and maintained in memory when a new ActiveReports file has been added. However, if a web site is bound to TFS, the **FileAttributes.xml** file is not maintained in memory. As a result, Visual Studio treats all the newly added files as normal code files.

Solution: You need to add the newly added files to **FileAttributes.xml** manually.

1. From the **Website** menu, select **Add New Item**.
2. Select **ActiveReports 6 (code-based) File** and click **OK**.
3. Close the project.

- Open the **FileAttributes.xml** file, add the new ActiveReports file, and set subtype to **Component**.

 **Note:** The FileAttributes.xml is located at C:\Documents and Settings\[username]\Local Settings\Application Data\Microsoft\WebsiteCache\[WebSite1]\ (Windows XP), or at C:\Users\[username]\AppData\Local\Microsoft\WebsiteCache\[WebSite1]\ (Windows 7).

XML code. Paste inside FileAttributes.xml

```
<?xml version="1.0" encoding="utf-16" ?>
<DesignTimeData>
  <File RelativeUrl="App_Code/NewActiveReport1.cs" subtype="Component" />
  <File RelativeUrl="Default.aspx.cs" subtype="ASPXCodeBehind"
codebehindowner="Default.aspx" />
  <File RelativeUrl="Default.aspx" subtype="ASPXCodeBehind" />
</DesignTimeData>
```

- Save the **FileAttributes.xml** file.
- Reopen the web site project.

The SystemNotSupportedException occurs when running ActiveReports with scripts on .NET Framework 4.0

Symptoms: The SystemNotSupportedException occurs when running ActiveReports with scripts on .NET Framework 4.0.

Cause: This exception occurs because of the CAS policy, which is obsolete in the .NET Framework 4.0.

Solution: There are two ways to resolve this issue.

- In the Solution Explorer, open the app.config file (for Windows Forms Applications) or the Web.config file (ASP.NET Web Applications) and add the following code.

(Windows Forms Applications) XML code. Paste inside the app.config file

```
<configuration>
  <runtime>
    <NetFx40_LegacySecurityPolicy enabled="true"/>
  </runtime>
</configuration>
```

(ASP.NET Web Applications) XML code. Paste inside the Web.config file

```
<system.web>
  <trust legacyCasModel="true"/>
</system.web>
```

- Or, set the **UseEvidence** property to False.

Microsoft Access OLE DB provider in a 64-bit system

Symptoms: Microsoft Access OLE DB provider, Microsoft.Jet.OLEDB.4.0 does not work on a 64-bit system.

Cause: In Visual Studio, by default, projects are set to use 32 bit or 64 bit, depending on the environment on which they are run. The Microsoft Access OLE DB provider, Microsoft.Jet.OLEDB.4.0, is not compatible with 64 bit, so it fails with Visual Studio on a 64-bit system.

Solution: To avoid this situation, change the project settings to use only 32 bit.

- With the project open in Visual Studio, from the **Project** menu, select Project Properties.
- In the page that appears, select the **Compile** tab in a VB project, or the **Build** tab in a C# project.
- Scroll to the bottom of the page and click the **Advanced Compile Options** button in VB, or skip this step in C#.
- Drop down the **Target CPU** list in VB, or **Platform target** in C#, (set to use AnyCPU by default) and select **x86**.
- Click **OK** to save the changes, or skip this step in C#.

Flash Viewer Troubleshooting

Swfobject undefined error

Symptoms: With the Flash viewer, my page throws a swfobject is undefined error.

Cause: The ActiveReports Handler Mappings are not set up correctly in IIS 7.0.

Solution: Update [ActiveReports Handler Mappings in IIS 7.0](#).

IOError while loading document. Error #2032

Symptoms: When running the Flash viewer in IIS, an error occurs with the following message: "IOError while loading document. Reason: Error #2032."

Cause: The ActiveReports Handler Mappings are not set up correctly in IIS or in the web.config file.

Solution: Update [ActiveReports Handler Mappings in IIS](#) or if that is already done, ensure that the [handlers are enabled](#) in your web.config file.

FireFox displays white pages

Symptoms: When running the Flash viewer in FireFox, displays white pages, but Internet Explorer renders reports correctly.

Cause: The height and width of the Flash viewer control is set to 100%. FireFox does not support this setting, so it does not resize the Flash Viewer at all.

Solution: Use cascading style sheets (CSS) to set the properties.

To use CSS in your Flash viewer ASPX

ASPX CSS code. Paste in the external CSS file

```
<style type="text/css">
html, body, #WebViewer1, #WebViewer1_controlDiv
{
    width: 100%;
    height: 100%;
    margin: 0;
}
</style>
```

To use an external CSS file

1. Assign the Flash viewer control a CSS class of **report-viewer**.
2. Add code like the following to the CSS file.

ASPX CSS code. Paste in the external CSS file

```
.report-viewer, .report-viewer div, .report-viewer object {height: 100%; width: 100%;}
```

Silverlight Viewer Troubleshooting

JScript error while using the Silverlight Viewer in Silverlight 5

Symptoms: When using the Silverlight Viewer in Silverlight 5, the JScript error may occur with the following message: "Unhandled Error in Silverlight Application The invocation of the constructor on type 'DataDynamics.ActiveReports.Viewer' that matches the specified binding constraints threw an exception."

Cause: The Silverlight Viewer is based on Silverlight 4, and it adds reference to System.Windows.Controls.dll (2.0.5.0). In Silverlight 5, adding the Silverlight Viewer control does not automatically add reference to System.Windows.Controls.dll to the project because the Silverlight Viewer adds reference to the Silverlight 4 version.

Solution: You should manually add reference to System.Windows.Controls.dll (in Silverlight 4 SDK or Silverlight 5 SDK) to the project.

Export Troubleshooting (Separate topic)

Parameters Troubleshooting

Error message appears in Fields list

Symptoms: An error message is displayed in the Fields list in the Report Explorer instead of the fields.

Cause: This is an expected error if no default value is given. If the field is a data type other than text, memo, or date/time in Access, the report will run normally.

Solution: To display the fields in the Fields list in the Report Explorer, supply a default value for the parameter in the Properties Window, or in the SQL query as below:

```
<%Name | PromptString | DefaultValue | DataType | PromptUser%>
```

Only the **Name** parameter is required. To use some, but not all, of the optional parameters, use all of the separator characters but with no text between one and the next for unused parameters. For example:

```
<%Name | | DefaultValue | |%>
```

An unhandled exception of type "System.Data..." occurs when the report is run

Symptoms: When the report is run, an exception like the following occurs: "An unhandled exception of type "System.Data.OleDb.OleDbException" occurred in system.data.dll"

Cause: If the field is a text, memo, or date/time data type in Access, the parameter syntax requires single quotes for text or memo fields, or pound signs for date/time fields. Please note that for different data sources, these requirements may differ.

Solution: To avoid the exception when the report is run against an Access database, use pound signs for date/time values, or single quotes for string values in your SQL query, for example:

```
#<%InvoiceDate | Choose invoice date: | 11/2/04 | D | True%>#
```

or

```
"<%Country | Country: | Germany | S | True%>"
```

User is prompted for parameters for subreports even though they are supplied by the main report

Symptoms: The parameter user interface pops up at run time asking for a value even though the main report is supplying the parameter values for the subreports.

Cause: The default value of the ShowParameterUI property of the report is True.

Solution: Set the ShowParameterUI property of the report to False. This can be done in the property grid or in code in the ReportStart event.

Print Troubleshooting

The printing thread dies before the report finishes printing

Symptoms: The printing thread dies before the report is printed.

Cause: If printing is done in a separate thread and the application is shut down right after the print call, the separate thread dies before the report is printed.

Solution: Set the usePrintingThread parameter of the Print() method to False to keep the printing on the same thread.

```
//C#
private void rptPrint_ReportEnd(object sender, System.EventArgs eArgs)
{
    this.Document.Print(false, false, false);
}
```

```
"Visual Basic
Private Sub rptPrint_ReportEnd(ByVal sender As Object, ByVal e As System.EventArgs) Handles
  MyBase.ReportEnd
  Me.Document.Print(False, False, False)
End Sub
```

The viewer shows the report on the wrong paper size

Symptoms: In the viewer, the report renders to a different paper size than the one specified.

Cause: ActiveReports polls the printer driver assigned to the report to check for clipping, margins, and paper sizes supported by the printer. If the paper size specified for the report is not supported by the printer, ActiveReports uses the printer's default paper size to render the report.

Solution: If the report is to be printed, the printer assigned to the report must support the paper size and margins. Please note that any changes to the print settings in code must be made in or before the **ReportStart** event. To use custom paper sizes not supported by the driver, set the **PrinterName** to an empty string to use the ActiveReports virtual print driver. This does not allow printing, but is recommended for reports that are only exported or viewed. This prevents Activerports from making a call to the default printer driver. Use the following code in the **ReportStart** event, or just before .Run is called.

C# code. Paste INSIDE the ReportStart event.

```
this.Document.Printer.PrinterName = '';
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Me.Document.Printer.PrinterName = ''
```

The PaperHeight and PaperWidth properties, which take a float value defined in inches, have no effect unless you set the PaperKind property to Custom. Here is some sample code which can be placed in the ReportStart event, or just before .Run.

C# code. Paste INSIDE the ReportStart event.

```
this.PageSettings.PaperKind = Drawing.Printing.PaperKind.Custom;
this.PageSettings.PaperHeight = 2; //sets the height to two inches
this.PageSettings.PaperWidth = 4; //sets the width to four inches
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Me.PageSettings.PaperKind = Drawing.Printing.PaperKind.Custom Me.PageSettings.PaperHeight
= 2 'sets the height to two inches Me.PageSettings.PaperWidth = 4 "sets the width to four
inches
```

Custom paper sizes do not work

Symptoms: Custom paper sizes do not work.

Cause: You can create more than one custom paper size, so setting only the **PaperKind** property is not enough to create a custom paper size.

Solution: In addition to setting the **PaperKind** property to **Custom**, you must also set the **PaperName** property to a unique string.

Blank pages are printed between report pages

Symptoms: Blank pages are printed between pages of the report.

Cause: This problem occurs when the **PrintWidth** plus the left and right margins exceeds the paper width. For example, if the paper size were set to A4, the PrintWidth plus the left and right margins should not exceed 8.27"; otherwise blank pages are printed. At run time, ActiveReports marks a page overflow by displaying a red line in the viewer at the position in which the breach has occurred.

Solution: Adjust the **PrintWidth** in the report designer using either the property grid or by dragging the right edge of the report. Page margins, height, and width can be adjusted either through the print properties dialog box in the **Report** menu

under **Settings**, or programmatically in the Report_Start event.

Memory Troubleshooting

Symptoms: ActiveReports is consuming too much memory and the CPU usage is at 100%.

The CPU usage always goes to 100% when using ActiveReports.

Cause: There are several reasons why too much memory may be consumed:

The report is not being disposed of properly

Cause: The report is not being disposed of properly. The *incorrect* syntax is as follows.

```
//C#  
rpt.Dispose();  
rpt=null;  
  
'Visual Basic.NET  
rpt.Dispose()  
rpt=Nothing
```

Solution: The correct syntax for disposing of a report is as follows.

```
//C#  
rpt.Document.Dispose();  
rpt.Dispose();  
rpt=null;  
  
'Visual Basic.NET  
rpt.Document.Dispose()  
rpt.Dispose()  
rpt=Nothing
```

Machine.Config MemoryLimit setting is insufficient

Cause: Large reports in an ASP.NET application can easily use up the 60% of memory allocated to the ASP.NET worker process by default, which produces an error. In Machine.Config, MemoryLimit specifies the maximum allowed memory size, as a percentage of total system memory, that the worker process can consume before ASP.NET launches a new process and reassigns existing requests.

Solution: Set the **CacheToDisk** property of the document to **True**.

This caches the report to disk instead of holding it in memory. This setting is also detected by the PDF Export, which follows suit, but any other exports still consume memory. Although it is not advised, the ASP.NET worker process memory allocation can also be changed in your Machine.Config file, which is located in a path like:

C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322\CONFIG\. Search the Machine.Config file for memoryLimit which is located in the processModel.

Report never finishes processing

Cause: In some cases, very large reports can consume so much memory that the report never finishes processing. Some of the things that can cause this include:

1. Many non-repeating images, or a high resolution repeating image
2. Instantiating a new instance of a subreport each time the format event of a section fires
3. Using a lot of subreports instead of grouping with joins in the SQL query
4. Pulling in all of the data when only a few fields are needed (e.g. **Select * from db** instead of **Select First, Last, Address from db**)

Solution: In cases where the report is too large to run any other way, the **CacheToDisk** property may be set to **True**. This property should only be used when there is no other way to run the report to completion. Before resorting to this method, please see the **Optimizing ActiveReports** topic.

Task manager indicates the current "working set" of the process

Cause: If inflated memory usage is seen in the Task Manager it is not necessarily in use by the code. Task manager indicates the current "working set" of the process and, upon request, other processes can gain access to that memory. It is managed by the Operating System.

Solution: For an example of some working set behavior anomalies (which are considered normal), create a WinForms application and run it. Look in Task Manager at the working set for that process (it should be several megabytes), then minimize and maximize the form and notice that the working set reclaims to <1MB. Obviously, the code was not using all that memory even though Task Manager showed that it was allocated to that process. Similarly, you'll see ASP.NET and other managed service processes continue to gradually grow their working set even though the managed code in that process is not using all of it. To see whether this is the case, try using the two lines of code below in a button Click event after running the project.

```
System.Diagnostics.Process pc = System.Diagnostics.Process.GetCurrentProcess();  
pc.MaxWorkingSet = pc.MinWorkingSet;
```

If that reclaims the memory then the Operating System trimmed the working set down to the minimum amount necessary and this indicates that the extra memory was not actually in use.



Note: According to Microsoft it is not necessary to call GC.Collect and it should be avoided. However, if calling GC.Collect reduces the memory leak, then this indicates that it is not a leak after all. A leak in managed code is caused by holding a reference to an object indefinitely. If ActiveReports is holding a reference to an object, then the object cannot be collected by the garbage collector.

WebViewer Troubleshooting

The WebViewer will not print without displaying the report

Symptoms: The WebViewer will not automatically print a report without displaying it.

Cause: Only the new FlashViewer ViewerType of the WebViewer offers this functionality.

Solution:

1. Set the **ViewerType** property to **FlashViewer**.
2. Expand the **FlashViewerOptions** property, and expand the **PrintOptions** subproperty.
3. Under the **PrintOptions** subproperty, set the **StartPrint** property to **True**.

The report is not getting updated with new data, or the page number stays the same

Symptoms: The WebViewer stays on the page number last viewed in the previous report when a user selects a new report or refreshes the current report, or new data does not display on refresh.

Cause: If the control is loaded in response to a client postback, the Report property does not run the specified report. Instead it uses a previously cached copy of the report's Document in the WebCache service to supply speedy responses to clients.

Solution: To force the client to use a new instance, call the **ClearCachedReport** method before setting the Report property.

The report in the HTML viewer type does not look exactly like the other viewer types

Symptoms: The report in the HTML viewer type does not look exactly like the other viewer types.

Cause: The HTML format is not WYSIWYG. It does not support the following items:

- Line control
- Control borders
- Shapes (other than filled rects)
- CrossSectionBox and CrossSectionLine controls
- Overlapping controls

Solution: Try to avoid using the above items in reports which are shown in HTML format.

The icons are missing on my WebViewer control

Symptoms: The icons are missing on my WebViewer control.

Cause: The httpHandlers in the Web.config file are missing or referencing the wrong version.

Solution: Ensure that the following HTTP Handler code is in the **Web.config** file and that the version is current.

```
***** ActiveReports HttpHandler Configuration *****
-->
    <add verb="" path="*.rpx" type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web, Version=6.0.0.5280, Culture=neutral, PublicKeyToken=cc496777c49a3ff"/>
    <add verb="" path="*.ActiveReport" type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler, ActiveReports.Web, Version=6.0.0.5280, Culture=neutral, PublicKeyToken=cc496777c49a3ff"/>
    <add verb="" path="*.ArCacheItem" type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler, ActiveReports.Web, Version=6.0.0.5280, Culture=neutral, PublicKeyToken=cc496777c49a3ff"/>
</httpHandlers>
```

"Error Creating Control - Webviewer"

Symptoms: "Error Creating Control - Webviewer" appears on the WebForm in place of the WebViewer control.

Cause: There is a version conflict within the project.

Solution:

1. Open the ASPX page and look in the **Source** view for a line that looks similar to the following and remove it:

```
<%@ Register TagPrefix="ActiveReportsWeb" Namespace="DataDynamics.ActiveReports.Web"
    Assembly="ActiveReports.Web, Version=6.0.0.5280, Culture=neutral, PublicKeyToken=c496777c49a3ff" %>
```

2. Right-click the ASPX page and select **View Code**. In the code view that appears, remove the following line from the Web Form Designer Generated Code:

```
Protected WithEvents WebViewer1 As DataDynamics.ActiveReports.Web.WebViewer
```

3. In the Solution Explorer, under References, right click and Remove the ActiveReports.Web reference.
4. Delete the control from the WebForm.
5. Add the WebViewer back to the WebForm.
6. Open the Web.config file in the solution, and scroll down to the httpHandlers tag near the bottom. There are three add verb tags on that line which indicate the Version. Update the version number.

Blank reports with the AcrobatReader viewer type on the production web server

Symptoms: In the WebViewer, reports render correctly with the HTML ViewerType but they show up blank with the AcrobatReader ViewerType on the production web server.

Cause: .ArCacheItem is not set up in your IIS extension mappings.

Solution:

1. From the Start menu, choose **Control Panel**, then **Administrative Tools**, then **Internet Information Services**.
2. Right-click your Default Web Site and choose **Properties**.
3. On the Home Directory tab, click the **Configuration** button.
4. On the Mapping tab, check the Extension column to see whether .ArCacheItem appears. If not, click **Add**.
5. In the Add/Edit Application Extension Mapping dialog that appears, click **Browse** and navigate to (Windows)\Microsoft.NET\Framework\v2.0.50727 or v3.0 or v3.5.

6. In the Open dialog, change **Files of type** to Dynamic Link libraries (*.dll).
7. Select **aspnet_isapi.dll** and click **Open**.
8. In the Extension textbox type **.ArCacheItem**.
9. Click the **Limit to** radio button and type **GET,HEAD,POST,DEBUG**.
10. Ensure that the **Script engine** check box is selected and the **Check that file exists** check box is cleared.
11. Click **OK**.

A configuration error occurs when deploying an application with the WebViewer on the Windows Azure Developer Portal

Symptoms: When an application with the WebViewer is deployed on the Windows Azure Developer Portal, a configuration error occurs.

Cause: The ActiveReports.PdfExport and ActiveReports.HtmlExport DLLs are not copied to the server at the application deployment.

Solution:

1. Add the ActiveReports.PdfExport and ActiveReports.HtmlExport DLLs to the WebRole project directory.
2. Ensure that **CopyLocal** is set to **True** for all ActiveReports DLLs in the project.
3. In Solution Explorer, open the Web.config file and add the following code to the <system.webServer> section:

XML code. Paste in the XML view of the Web.config file inside the system.webServer section.

```
<handlers>
  <add name="RPX" verb="*" path="*.rpx"
  type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
  Version=6.1.3027.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
  <add name="ActiveReport" verb="*" path="*.ActiveReport"
  type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler,
  ActiveReports.Web, Version=6.1.3027.0, Culture=neutral,
  PublicKeyToken=cc4967777c49a3ff" />
  <add name="Cache" verb="*" path="*.ArCacheItem"
  type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler,
  ActiveReports.Web, Version=6.1.3027.0, Culture=neutral,
  PublicKeyToken=cc4967777c49a3ff" />
</handlers>
```

 **Note:** You should update the Version and PublicKeyToken values to reflect the current version of ActiveReports installed on your machine.

You can find the Version and PublicKeyToken values in the Global Assembly Cache (GAC),
C:\WINDOWS\ASSEMBLY.

A configuration error occurs when running a Windows Azure application with the WebViewer

Symptoms: A configuration error occurs when running a Windows Azure application with the WebViewer.

Cause: Some handlers in the Web.config file are not properly configured.

Solution:

1. In Solution Explorer, open the Web.config file and add the following code to the <system.webServer> section:


XML code. Paste in the XML view of the Web.config file inside the system.webServer section.

```
<handlers>
  <add name="RpxHandler" verb="*" path="*.rpx"
  type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
  Version=6.1.3031.1, Culture=neutral, PublicKeyToken=cc4967777c49a3ff"
  resourceType="Unspecified" />
```

```

    <add name="CompiledReportHandler" verb="*" path="*.ActiveReport"
    type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler,
    ActiveReports.Web, Version=6.1.3031.1, Culture=neutral,
    PublicKeyToken=cc4967777c49a3ff" resourceType="Unspecified" />
    <add name="WebCacheAccessHandler" verb="*" path="*.ArCacheItem"
    type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler,
    ActiveReports.Web, Version=6.1.3031.1, Culture=neutral,
    PublicKeyToken=cc4967777c49a3ff" resourceType="Unspecified" />
  </handlers>

```

 **Note:** You should update the Version and PublicKeyToken values to reflect the current version of ActiveReports installed on your machine.

You can find the Version and PublicKeyToken values in the Global Assembly Cache (GAC),
C:\WINDOWS\ASSEMBLY.

2. In the Web.config file, remove or comment out the following code from the <system.web> section:

XML code. Remove from the system.web section in the Web.config file.

```

<httpHandlers>
  <add verb="*" path="*.rpx"
  type="DataDynamics.ActiveReports.Web.Handlers.RpxHandler, ActiveReports.Web,
  Version=6.1.3011.0, Culture=neutral, PublicKeyToken=cc4967777c49a3ff" />
  <add verb="*" path="*.ActiveReport"
  type="DataDynamics.ActiveReports.Web.Handlers.CompiledReportHandler,
  ActiveReports.Web, Version=6.1.3011.0, Culture=neutral,
  PublicKeyToken=cc4967777c49a3ff" />
  <add verb="*" path="*.ArCacheItem"
  type="DataDynamics.ActiveReports.Web.Handlers.WebCacheAccessHandler,
  ActiveReports.Web, Version=6.1.3011.0, Culture=neutral,
  PublicKeyToken=cc4967777c49a3ff" />
</httpHandlers>

```

The DocumentLoadException occurs when deploying a project that contains the WebViewer with an RDF file on the Windows Azure Developer Portal

Symptoms: The DocumentLoadException occurs when deploying a project that contains the WebViewer with an RDF file on the Windows Azure Developer Portal.

Cause: An RDF file is not copied to the server at the report deployment.

Solution: In Solution Explorer, you should select an RDF file and set its **BuildAction** property to Content and its **Copy to Output Directory** property to Copy always.

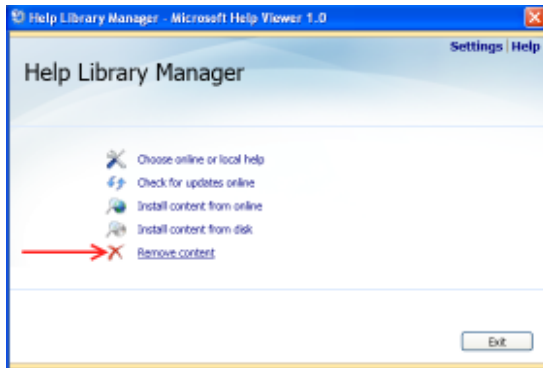
Help Troubleshooting

ActiveReports 6 Help is not updated in Visual Studio 2010 after the new service pack installation

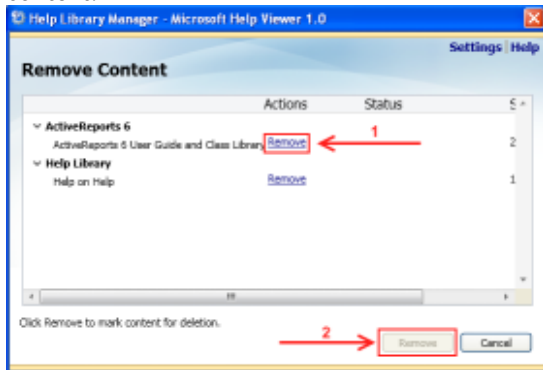
Symptoms: ActiveReports 6 Help is not updated in Visual Studio 2010 after the new service pack installation.

Solution:

1. Open the **Help Library Manager** by selecting **Manage Help Settings** in the **Microsoft Visual Studio 2010/Visual Studio Tools** folder, or in the **Help** menu of Visual Studio 2010.
2. Select **Remove content** in the **Help Library Manager**.



3. Click the **Remove** action next to the content title and then click **Remove** to remove the old ActiveReports 6 Help content.



4. Install ActiveReports 6 Help as described in **Installing ActiveReports 6 Help into Visual Studio 2010**.

Pressing the F1 key does not open ActiveReports 6 Help in Visual Studio 2010

Symptoms: I have installed ActiveReports 6 Help as described in **Installing ActiveReports6 Help into Visual Studio 2010** but pressing the F1 key does not open the documentation file in Visual Studio 2010.

Solution: Make sure that you have restarted the Help Library Agent. If restarting the Help Library Agent does not resolve this issue, please contact our support team: powersupport@grapecity.com.

Export Troubleshooting

If you run into an issue while using ActiveReports exports, you will probably find the solution within this section. Click any short description below to drop down the symptoms, cause, and solution.

General Export Troubleshooting

Specified cast not valid exception

Symptoms: The "Specified Cast Is Not Valid" exception fires.

Cause: There is a mix of old and new dlls in the references.

Solution:

1. Remove all ActiveReports 6 references from the project.
2. Add the ActiveReports 6 references necessary to your project, making sure to use only references to assemblies from the latest build installed on your machine.

Excel Export Troubleshooting

Extra columns in exported spreadsheet

Symptoms: Extra columns are showing up in the Excel export.

Cause: Controls don't have the same Top and Height properties. When the Excel export runs, it divides the report into rows and columns based on the borders of the controls, much like a grid. Controls that are not aligned across the report cause the Excel export to produce more columns to accommodate them.

Solution: Use the new SnapLines feature to help align the controls. To remedy the situation, set as many controls with the same Left property values and the same Top property values as you can, aligning them as if they were in a grid. This reduces the number of columns exported. Also, setting the RemoveVerticalSpace or UseCellMerging property to True, or setting the MinColumnWidth property equal to the width of the narrowest cell may help.

Extremely long reports do not export

Symptoms: Extremely long reports do not export to Excel.

Cause: The maximum number of rows which can be exported to MS Excel version 8.0 or higher is 65,536, while older versions of Excel (4.0, 5.0 and 7.0) had a limit of 16,384 rows.

Here are other specifications for Excel version 9.0 which may affect your export:

- Maximum worksheet size: 65,536 rows by 256 columns
- Maximum column width: 255 characters
- Maximum row height: 409 points
- Maximum length of cell contents (text): 32,767 characters. Only 1,024 display in a cell; all 32,767 display in the formula bar.

Solution: Use the Export(document,filePath,pageRange) or Export(document,outputStream,pageRange) method to export ranges of pages into separate Excel documents.

Export fails sporadically in memory stream

Symptoms: When using a memory stream, the Excel export sporadically fails.

Cause: Internet Explorer requires a "content-disposition" header in the response.

Solution: Use code like the following before creating the export.

Paste this code

```
Response.ContentType = "application/x-msexcel";  
Response.AddHeader("content-disposition", "attachment; filename=MyXLS.XLS");  
Response.AddHeader("content-disposition", "inline; filename=MyXLS.xls");
```

The export does not look like the original

Symptoms: The exported Excel file does not look exactly like the original report.

Cause: The Excel export is not WYSIWYG. It does not support the following items:

- Line control
- Borders on controls with angled text
- Shapes (other than filled rects)
- Overlapping controls

Solution: Try to avoid using the above items in reports which will be exported to Excel.

HTML Export Troubleshooting

Invalid syntax error

Symptoms: Internet Explorer displays "Invalid syntax error" on the title bar, "The page cannot be displayed" in the body and the URL is prepended with *mhtml:* in the address bar.

Cause: This is caused by [Microsoft Hotfix Q330994](#). Microsoft addressed an Outlook security flaw that used the MHT extension and broke the streaming HTML functionality.

Solution: Append the *mht* extension to the file when you link to it. For example,

```
SortSample.aspx?RunReport=true
```

Would be changed to the following.

```
SortSample.aspx?RunReport=true f=temp.mht
```

The export does not look like the original

Symptoms: The exported HTML file does not look exactly like the original report.

Cause: The HTML export is not WYSIWYG. It does not support the following items:

- Line control
- Control borders
- Shapes (other than filled rects)
- Overlapping controls

Solution: Try to avoid using the above items in reports which will be exported to HTML.

Exported vertical text in the Textbox and Label controls does not look like the original

Symptoms: Vertical text in the Textbox and Label controls in the exported HTML file does not look like the original report.

Cause: The export of vertical text to HTML is supported just for the DynamicHtml output type and works with Microsoft Internet Explorer only.

Solution: Set the **OutputType** property of the HTMLExport object to DynamicHtml and use Microsoft Internet Explorer to open the exported HTML file.

PDF Export Troubleshooting

Barcodes do not scan and margins expand

Symptoms: Barcodes in printed PDF exports do not scan and the page margins appear larger than the ones in the original report.

Cause: The Adobe Acrobat reader has a default setting in the Print dialog which tells it to scale down large pages. The reader views ActiveReports margins as part of the document, and renders them inside the new margins it creates. This is normally not noticeable unless barcodes are used or the PDF printout is held up next to a printout of the original report.

Solution:

For Acrobat 6, 7, or 8, change the following options in the Print dialog under Page Handling:

1. Set Page Scaling to None.
2. Clear Auto-Rotate and Center.

For Acrobat 5, clear the following options in the Print dialog:

1. Shrink oversized pages to paper size
2. Expand small pages to paper size
3. Auto-rotate and center pages

The WebViewer shows blank PDF reports

Symptoms: In the WebViewer, reports render correctly with the HTML viewer type but they show up blank with the AcrobatReader viewer type on the production Web server.

Cause: .ArCacheItem is not set up in your IIS extension mappings. Visual Studio now has the built-in Cassini Web

server, so this problem does not show up during testing. However, this still needs to be set up for production.

Solution:

1. On the production Web server, from the Start menu, choose All Programs, Administrative Tools, Internet Information Services.
2. Right-click your Default Web Site and choose Properties.
3. On the Home Directory tab, click the Configuration button.
4. On the Mapping tab, check the Extension column to see whether .ArCacheItem appears there.
5. If .ArCacheItem does not appear click the Add button.
6. Browse to C:\WINDOWS\Microsoft.NET\Framework\v2.x.xxxx or C:\WINNT\Microsoft.NET\Framework\v2.x.xxxx (or v3.x.xxxx).
7. Change Files of type: to Dynamic Link libraries (*.dll).
8. Choose the aspnet_isapi.dll and click Open.
9. In the Extension textbox type ".ArCacheItem".
10. Click the Limit to: radio button and type "GET,HEAD,POST,DEBUG".
11. Make sure that the "Script engine" checkbox is checked and the "Check that file exists" checkbox is not checked.
12. Click OK.

Export fails in the FIPS-compliant environment if CacheToDisk is used

Symptoms: The "Object reference not set to an instance of an object" exception fires.

Cause: The IsolatedStorage class is not FIPS-compliant.

Solution: You should disable the **CacheToDisk ('CacheToDisk Property' in the on-line documentation)** property or set the **CacheToDiskLocation ('CacheToDiskLocation Property' in the on-line documentation)** property, using the code like the following:

C#

```
rptTest ar = new rptTest();
ar.Document.CacheToDisk = true;
ar.Document.CacheToDiskLocation = "c:\\\";
```

Visual Basic

```
Dim ar As New rptTest()
ar.Document.CacheToDisk = true
ar.Document.CacheToDiskLocation = "c:\"
```

The ArgumentOutOfRangeException exception occurs when using a custom font factory in an application that exports a report and runs it on Azure Fabric

Symptoms: The ArgumentOutOfRangeException exception occurs when using a **custom font factory** in an application that exports a report and runs it on Windows Azure Development Fabric.

Cause: Fonts files are not copied to the server at the application deployment.

Solution: You should set **BuildAction** to **Content** and **Copy to Output Directory** to **Copy always** for all Fonts in the project Fonts folder.

A configuration error occurs when deploying an application with a custom font factory in a Windows Azure project

Symptoms: A configuration error occurs when an application with a custom font factory is deployed in a Windows Azure project.

Cause: The ActiveReports.Web.dll is not present in the project and some ActiveReports DLLs are not copied to the server at the application deployment.

Solution: Add the ActiveReports.Web.dll to the project and ensure that **CopyLocal** is set to **True** for

all ActiveReports DLLs in the project.

The ConfigurationErrorsException error occurs when exporting an application with a custom font factory in Azure worker role project

Symptoms: When exporting an application with a custom font factory in Azure worker role project, the ConfigurationErrorsException error occurs.

Cause: Azure worker role does not support a virtual path.

Solution: You should specify the absolute path to the **Font** folder as `<AddFolder Path="Fonts" Recurse="true"/>` in the **font factory section** of app.config file.

RTF Export Troubleshooting

The export does not look like the original

Symptoms: The exported RTF file does not look exactly like the original report.

Cause: The RTF export is not WYSIWYG, and is based on WordPad rather than Word, so there are some limitations. The following items are not supported in the RTF export.

- Line controls
- Backcolor
- Shape controls
- Overlapping controls
- Control borders (except for borders on the RichTextBox control, which are supported)
- Angled text

Solution: Try to avoid using the above items in reports which will be exported to RTF.

Text Export Troubleshooting

The export does not look like the original

Symptoms: The exported text file does not look exactly like the original report.

Cause: The text export is not WYSIWYG. It is limited to plain, unformatted text, or plain text with encoding for foreign language support.

Solution: Only use the Text export when the output can be plain text with no formatting.

TIFF Export Troubleshooting

The export is completely black

Symptoms: Reports exported to TIFF format with the RLE, CCITT3 or CCITT4 compression scheme are rendering black.

Cause: The sections in the reports have a BackColor and are not dithered.

Solution: Set the Dither property for the export object to True, or set the BackColor property of each section to Transparent.

The export has extra white space at the bottom of the page

Symptoms: Some reports are exported to TIFF with extra white space at the bottom of the page.

Cause: The default value for CompressionScheme is CCITT3. According to CCITT3 standards, the page size is 1728 x 2376 pixels and the resolution is 200 x 196 DPI, as required by some fax machines. If the page is smaller than this standard, then it is scaled up to CCITT3 standards, which can cause extra white space at the bottom.

Solution: To get an accurate page size, set the CompressionScheme to CompressionScheme.Rle (Run length encoding).